The Transactor

The Tech/News Journal For Commodore Computers Vol. 4.

April 1983

Issue 03
$2.95

VIC 20 / C64
BASIC LINE
LABELLER

VIC 20
SCREEN
CENTERING

C64
PROGRAMMABLE
CHAR. EDITOR

CATSTRAPOLATOR
CATALOG
EXTRAPOLATOR

C64 TO 8023P
HI-RES SCREEN
DUMPER

MOVING THE
C64
CHARACTER SET

BUTTERFIELD:
ON 16 BITS

MORE NOTES
ON SID SOUND

C64
STRING THING

C64
BASIC AID

WORDPRO
TRICKS

MANAGER
COLUMN

J. MOSTACCI

Richvale Telecommunications

10610 BAYVIEW (Bayview Plaza)
RICHMOND HILL, ONTARIO, CANADA L4C 3N8
(416) 884-4165

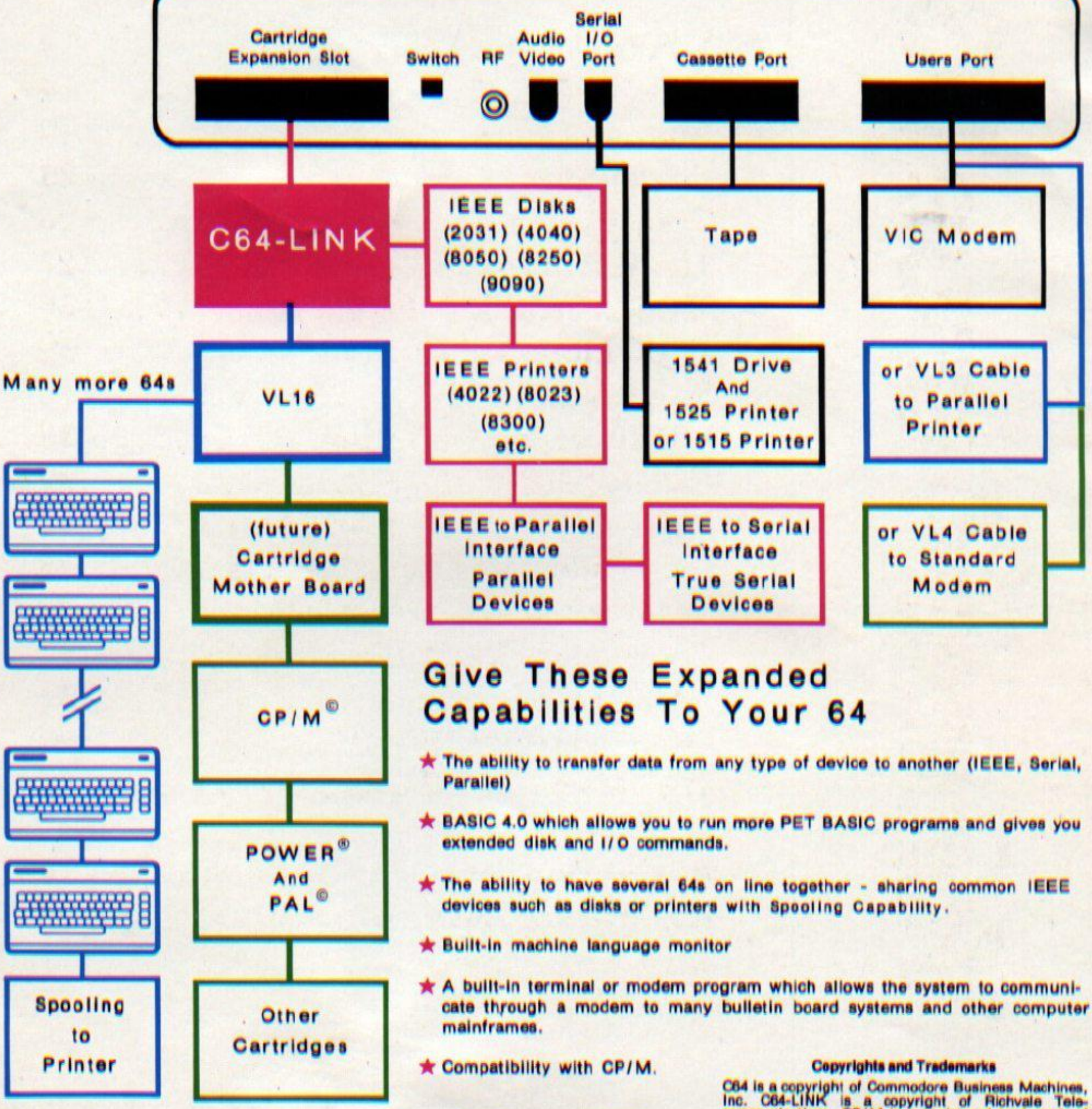
$185 Canadian

C64-LINK®

The Smart 64

Also available for VIC 20

RTC RTC



Give These Expanded Capabilities To Your 64

★ The ability to transfer data from any type of device to another (IEEE, Serial, Parallel)

★ BASIC 4.0 which allows you to run more PET BASIC programs and gives you extended disk and I/O commands.

★ The ability to have several 64s on line together - sharing common IEEE devices such as disks or printers with Spooling Capability.

★ Built-in machine language monitor

★ A built-in terminal or modem program which allows the system to communicate through a modem to many bulletin board systems and other computer mainframes.

★ Compatibility with CP/M.

Contact your local Commodore dealer or RTC.

Copyrights and Trademarks

C64 is a copyright of Commodore Business Machines, Inc. C64-LINK is a copyright of Richvale Telecommunications. CP/M is a registered trademark of Digital Research. POWER is a trademark of Professional Software. PAL is a copyright of Brad Templeton.

The Transactor

Volume 4 Issue 03

Editorial 3
News BRK 4
Bits and Pieces 18
CompuKinks 25
Transbloopors 26
The WordPro Book of Tricks 27
The MANAGER Column 30
TPUG Library Filenames 32
Sweet Sixteen 34
VIC 20 Screen Centering 36
Catstrapolator 39
Making Friends With SID, Part 2 42
Commodore 64 Character Sets 46
Programmable Character Editor 50
Commodore 64 Hi–Res Dumper 52
BASIC Line Labeling 54
Commodore 64 Tiny–Aid 60
1982 Magazine Bibliography 67

Advertising Section 74
Advertising Index 80

The Transactor
The Tech/News Journal For Commodore Computers

Editor
Karl J. H. Hildon

Advertising Manager
Kelly George
416 826 1662

Contributing Writers
Dave Berezowski
Jim Butterfield
Gord Campbell
Mike Donegan
Donna Green
Paul Higginbottom
Dave Hook
Eike Kaiser
Peter Lear
Bill MacLean
John Stoveken
Don White

Production
Attic Typesetting Ltd.

Printing
Squire & Carter Ltd.

Artwork / Cover Art
John Mostacci

Program Listings In The Transactor

All programs listed in The Transactor will appear as they would on your screen in Upper/Lower case mode. To clarify two potential character mix-ups, zeroes will appear as '0' and the letter "o" will of course be in lower case. Secondly, the lower case L ('l') has a flat top as opposed to the number 1 which has an angled top.

Many programs will contain reverse video characters that represent cursor movements, colours, or function keys. These will also be shown exactly as they would appear on your screen, but they're listed here for reference.

Occasionally programs will contain lines that show consecutive spaces. Often the number of spaces you insert will not be critical to correct operation of the program. When it is, the required number of spaces will be shown. For example:

print" flush right" – would be shown as – print" [space10]flush right"

Cursor Characters For PET / CBM / VIC / 64

Down	– **q**	**Insert**	– **I**
Up	– **Q**	**Delete**	– **T**
Right	– **]**	**Clear Scrn**	– **S**
Left	– [Lft]	**Home**	– **s**
RVS	– **r**	**STOP**	– **c**
RVS Off	– **R**		

Colour Characters For VIC / 64

Black	– **P**	**Orange**	– **A**
White	– **e**	**Brown**	– **U**
Red	– **£**	**Lt. Red**	– **V**
Cyan	– [Cyn]	**Grey 1**	– **W**
Purple	– [Pur]	**Grey 2**	– **X**
Green	– **t**	**Lt. Green**	– **Y**
Blue	– **←**	**Lt. Blue**	– **Z**
Yellow	– [Yel]	**Grey 3**	– [Gr3]

Function Keys For VIC / 64

F1	– **E**	**F5**	– **G**
F2	– **I**	**F6**	– **K**
F3	– **F**	**F7**	– **H**
F4	– **J**	**F8**	– **L**

The Transactor is published bi-monthly, 6 times per year, by Canadian Micro Distributors, Limited. It is in no way connected with Commodore Business Machines Ltd. or Commodore Incorporated. Commodore and Commodore product names (PET, CBM, VIC, MAX, 64) are registered trademarks of Commodore Inc.

Volume 4 Subscriptions: Canada $15 Cdn
U.S.A. $15 US.
All other $18 US.

Second Class Mail
Permit Pending

Back issues are still available for Volumes 1, 2, & 3. Best of Volume 1: $10 Cdn., U.S.A. $10 US., all other $12 US. Best of Volume 2: $15 Cdn., U.S.A $17 US., all other $19 US. Volume 3: $15 Cdn., U.S.A $17 US., all other $19 US.

Volume 4 quantity orders: subtract 35% on orders of 10 or more.

Send all subscriptions to: The Transactor, Subscriptions Department, 500 Steeles Avenue, Milton, Ontario, Canada, L9T 3P7, 416 878 7277.

Want to advertise a product or service? Call or write for more information.

Editorial contributions are always welcome and will appear in the issue immediately following receipt. Preferred media is 2031, 4040, 8050, or 8250 diskettes with WordPro, WordCraft, Superscript, or SEQ text files. Program listings over 25 lines should be provided on disk or tape. Manuscripts should be typewritten, double spaced, with special characters or formats clearly marked. Photos of authors or equipment, and illustrations will be included with articles depending on quality. Diskettes, tapes and/or photos will be returned on request.

All material accepted becomes the property of The Transactor, until it is published. Once released, Authors may re-submit articles to other publications at their own discretion. Other magazines, of any nature, are invited to copy material from The Transactor, provided that credit is given to the Author (when applicable) AND The Transactor.

The opinions expressed in contributed articles are not necessarily those of The Transactor. Although accuracy is a major objective, The Transactor cannot assume liability for errors in articles or programs.

From The Editor's Desk

Make Extra CA$H from The Transactor

As of this issue, **The Transactor** will pay for all material accepted for publication. To start, the rates will be $20 per typeset page up to a maximum of $60.

As The Transactor grows, so will our remuneration, and of course the maximum will be lifted. However, research on attention span shows that dwelling on one subject too long can actually be worse that not long enough. Although you should attempt to include every detail, a short article that gets read is better than a long one that doesn't.

In general, articles should span no more than about 7 typewritten pages including program listings. With so much material being released, long drawn-out programs rarely get entered – potential error (publisher or during entry) and the fact that they can be obtained from most user clubs, make them almost unworthy of publication. You have to admit, printing a program 15,000 times is somewhat wasteful if only a handful of readers bother with it. Naturally, there will be programs of great value to many readers that deserve extra space, and they will be considered. But, as a rule, try to keep programs to 180 lines (3 pages) or less.

On the other hand we have short articles and/or programs which nearly always get read. I can't help but read a one paragraph blurb regardless of the subject. Subroutines, short utilities, and especially "one-liners" and novelty programs often get entered merely because of the time it *doesn't* take. Further, errors are easy to spot and improvements are easily slipped in without mucking up some other part of the program you may not immediately notice.

Make Me Laugh, Anytime.

Are you an artist? Or how 'bout a state-of-the-art comedian? Or both? If you have a cartoon dealing with some aspect of computing, The Transactor will pay you for them! They can be single frames with captions, multi-frame with dialogue, or for anyone interested, a continuing series like "The Adventures of Rodney Rambit, and His Fearless Companion, Chip Silicon". We'll leave it up to your imagination, but if it's good for a laugh, it's good for cash!

Each single frame with caption is worth $10. Send in three at once and get $40. Quality is important though. If your caption is one that'll "bring the house down" but your artwork makes it hard to avoid insulting you, we'll still pay you for them but at half the regular rate. We'll use the other half to pay a professional cartoonist, and both you and the artist will be credited. When The Transactor grows in size, so will the cartoon section, hopefully to around 5 or 6 pages full each issue. I think you'll agree, there's nothing better than a good howl before getting down to business!

We're still working on all the details and a writers package will be available soon that will be sent to all our previous writers and anyone else on request. The package will include a contract, technical specs (ie. for cartoon sizes), and a coupon good for a free subscription or renewal. If you have a submission ready now, send it in anyways and we'll use the blower to get in touch.

Perpetual improvement plan phase two now engaged!

Sincerely,

Karl J. H. Hildon
Editor

News BRK

The Transactor loves Press Releases.

Reference Issue Postponed

In our last issue (January 1983) we announced that this issue would be the Reference Issue containing handy charts and tables and other useful material of the kind that gets frequent perusal. However, we had hoped that Commodore's new third generation machines (the C and B/BX) would be released in time for us to include their memory maps, etc. Such was not the case so we've decided to hold off until this can be obtained. These and other new CBM machines are just about ready for market so the Reference Issue should be along soon (Issue 04 or 05).

One article planned for the Reference Issue was Don White's 1982 PET Bibliography, listing Commodore related material in other magazines published in 1982. To release this any later would be somewhat untimely so we've included it here in Issue 03.

On The Local Scene. . .

New Computer Oriented Book Store Opens in Toronto

The Computer Book & Supply Centre, Toronto's first specialized computer book retail store, held their Grand Opening March 30. They carry a wide range of titles, from introductory tutorials to advanced programming, and they invite special orders for obscure, hard-to-find publications. Already they offer more than 40 magazines plus other computer supplies including diskettes, printer paper and continuous forms/labels, printwheels, ribbons, and more.

The Computer Book & Supply Centre
253 Eglinton Avenue West
(south side, just east of Avenue Road)
416 489 3625
Open Mon–Sat – 10 am to 6 pm
Fri – 10 am to 9 pm

Computing NOW!

From the publishers ETI (Electronics Today Int'l) comes a new magazine called Computing NOW!. The magazine deals with all brands of micros and other facets of the micro industry.

Subscriptions are $18.95 with the option cancel at any time with refund of unused portion. Write to:

Computing NOW!
25 Overlea Blvd. Unit 6
Toronto, Ontario
M4H 9Z9

TVOverwhelming.

Last season TVOntario challenged Ontarians to forgo traditional passive viewing and get involved with a bold new television experience – one that combined print and electronic media.

That challenge was met head on!

Ten thousand registrants (as many as TVO could accommodate) in the *TVO Academy on Computers in Education* are now learning how to operate a computer, understand how it functions, and develop simple computer programs in the comfort of their own homes.

To meet the overwhelming response and accommodate registrant overflow, TVO will repeat this step–by–step orientation course in microcomputers Monday and Tuesday nights at 8:00 pm EDT for six weeks beginning May 2. (the original course was 12 weeks long on Wednesdays)

The TVO Academy on Computers in Education consists of *Bits and Bytes* (a 12–part TV series starring Billy Van and Luba Goy), study guides, a variety of readings, multiple choice questionaires, personalized correspondence, topical newsletters, and contributions from a network of local experts.

Each half–hour broadcast in *Bits and Bytes* is followed by *The Academy with Jack Livesley: On Computers* at 8:30 pm. Host Jack Livesley's resident guest expert for the series is none other than the ever present and well–known writer, programmer, lecturer, self–*un*proclaimed philosopher, jet–set world traveller, and The Transactor centerfold of the year. . . Jim Butterfield, *Super Guru.* [I just *had* to edit that part].

A fee of $59.00 gives a registrant a hands–on manual – a stand–alone aid for people with access to certain types of PET, Apple, or TRS–80 microcomputers – and several samples of computer programs to guide them through the operation and use of a computer. (without the manual, $53.00)

Both hour–long Monday and Tuesday night presentations will be repeated the following Saturday beginning May 7 from 2:30 to 4:30.

Editor's Note

I had the pleasure of attending a taping with Jim at TVO studios in Toronto. Very impressed. The content is well thought out and the show gets the full treatment. TVO research staff and other technical consultants for the show are present during taping for any last minute details; control room technicians are well organized and precise; make–up staff are standing by just off camera; and the camera crew is bang on. Combined with the devoted stars of the show, I felt an atmosphere of genuine concern for quality.

Of course, the show is not intended for veterans or even the moderately experienced user. But for those just getting into micros and programming, it's an excellent start.

On the set at TVO.

Events & Exhibitions

Second Annual TPUG Conference

The Toronto PET Users Group is holding its second annual club conference at the George Brown Casa Loma Campus in Toronto (same location as last year). Show dates are May 14th and 15th, Saturday from 11 am to 6:00 and Sunday from 11 to 4:30.

Activities include a disk copying marathon (1500+ programs) where attendees can leave blank disks at various copy stations and collect them at a later time. Concurrent seminars by invited speakers such as Jim Strasma, Frank Covitz and Cliff Ashcraft on music, Willi Cusche on KMM Pascal, Greg Yob, Lorne Wright (MICRO Magazine), and Jim Butterfield will hold a machine language workshop. Several exhibitors of hardware, software, and accessories will be present, and a traders corner for used equipment will surely attract some attention.

Last years conference was indeed a success, but now, with some accrued experience, this years' event should be even more enjoyable. Your editor will be there, however, I don't know yet if I'll be participating or just an observer. For more information contact:

Chris Bennett
TPUG Corresponding Secretary,
381 Lawrence Avenue West
Toronto, Ontario
M5M 1B9 416 782 9252

Editor's Note

The above number actually belongs to TPUG's new business office. Back in the winter of '78, TPUG held its first meeting in the basement party room of the condominium of founder Lyman Duggan – 22 people attended. Now with over 5000 members world wide, TPUG has full-time TOR-PET and business staff, a PR department, and regular business meetings where members are invited to attend. It shouldn't be long before executive elections are scheduled, but with everyone doing such a fine job, I can't imagine anyone being replaced.

TPUG is now the largest computer club in the world – yearly memberships are $30 for those attending regular meetings, $20 for students. Associate memberships are available for those not attending meetings at $20, overseas $30. Great value for money. When submitting applications, TPUG asks that you include a first AND last name, and postal code clearly identified. [same here. – Ed.]

First Annual Computer Fair

The Computer Fair, Canada's first national microcomputer show, is being held June 23rd to 26th 1983 at the International Center in Toronto.

All facets of this rapidly developing industry will be present: designers, manufacturers, retailers, users, and more. Hardware, software and applications will be featured in both the exhibit and seminar areas.

In addition, for the first time in any computer show, education, training and career opportunities will be an integral feature.

"Computer Career Opportunities" gives you the opportunity to present your institute to the public and encourage them to speak to your representatives face to face, discuss curriculum, and obtain literature about your courses.

An expected 50,000 visitors – business professionals, doctors, teachers, students, entrepreneurs, and generally interested consumers – will attend the four-day event to absorb information, and capture the excitement of this new and quickly expanding industry and potential job market.

As space is limited, we advise you to reserve early. Already there are over 100 exhibitors registered, all in the micro field.

Show hours are 10 am. to 10:30 pm, Thursday to Saturday, Sunday 11 to 6. Admission is $5, students and seniors $3. For more information, contact:

Gail Park
Computer Career Opportunities
2282 Queen Street East
Toronto, Ontario
M4E 1G6 416 690 9666

Editor's Note

I wouldn't miss this show for anything! It promises to be more compact than CCS and no wading through tons of mainframes, photocopiers, and other junk. This is the first of what looks like the most prospective annual micro show in Canada. For those outside Toronto, the International Center is right near the airport and there's lots of free parking and nighttime entertainment. Next year's fair is slated for May.

MidniteXPress Conference Set For June

The MidniteXPress (formerly Midnite Software Gazette/ The PAPER) is holding a week long Commodore computer conference at Lincoln College. Jim Butterfield, Jim Strasma, yours truly, and several other "famous" personalities will be there to conduct various seminars.

Midnite would like to hear from those planning to attend that can bring extra equipment. They're also offering a registration option that includes a 20 or 64. For details:

Midnite Conference
1280 Richland Avenue
Lincoln, IL
62656 217 735 2703

Waterloo University Seminar/Workshops

WATSOFT Products Inc. has been offering a variety of seminar/workshops including hardware design, interfacing techniques, structured programming and program design, machine language programming, and software portability. Each workshop is 3 consecutive days of intensive study conducted by Waterloo Professors and/or research staff at U of W's Computer Systems Group. The courses would be of particular interest to Computer Science, Math, Business, and Technology teachers, but other professionals would most certainly benefit.

The workshops continue throughout the year. For information on schedules, fees, location, etc., contact:

Greg Garrison
Seminar Co-ordinator
WATSOFT Products Inc.
158 University Avenue West
Waterloo, Ontario
N2L 3E9 519 886 3700
Telex 06 - 955458

Computers in Education '83

CE '83 is the third annual conference and summer institute for educators. Dates are June 20th through to July 15th, 1983, at Rutgers - State University of New Jersey.

This year's theme is "Necessary Direction for Computer Education: Navigational Aids for the 80's". The focus of the conference centers on microcomputers and other new information technologies and their impact on education at elementary, secondary, and college levels.

Among the participants are Jim Butterfield, David H. Ahl, Ludwig Braun, Trudy Van Buskirk, Don Whitewood, Wes Graham, and about 60 others. Based on past conferences, attendance this year is expected around 2000.

For a list of details write or call:

Dr. Mitchell E. Batoff
Director CE '83
Institute for Professional Development
245 Nassau Street, Suite D
Princeton, New Jersey
08540 609 924 8333

TERC Summer Computer Seminars

TERC (*T*echnical *E*ducation *R*esearch *C*enters) announces its second annual Summer 1983 Workshop Series, Microcomputers in Education, to be held at the TERC offices in Cambridge, MA. These four-day workshops will provide intensive training in a variety of topics for teachers and administrators at all levels.

The schedule for the summer series is:

June 20–23	Microcomputers in the Science Lab
July 5–8	Software Development Workshop
July 11–22	Special Summer Institute (see below)
July 25–28	Logo
August 1–4	Simulations
August 8–11	Pascal
August 15–18	Logo

The Summer Institute in Math, Science, and Computer Literacy will be held at Trinity College in the Green Mountains of Vermont. This workshop is offered as a one or two week session. Topics include:

Week 1, July 11–15
- Micros in Elementary Math & Science
- Logo
- BASIC
- Computer Literacy & Awareness

Week 2, July 19–22
- Micros in Algebra & Geometry
- Micros in Trigonometry & Calculus
- Math & Science Software Tools
- Micros in Natural Sciences
- Micros in the Physical Sciences

Morning and afternoon sessions offer maximum participant schedule flexibility. Enrollment is limited so early registration is advised. Special outdoor activities in beautiful Vermont are available for spouses, children, and participants. For more:

TERC
8 Eliot Street
Cambridge, MA
02138 617 547 3890

The Aftermarket

CompuServe Subscriptions

CompuServe, one of North America's leading large-scale timeshare networks, is now selling access to the service through computer retailers and major retail chains.

CIS (*C*ompuServe *I*nformation *S*ervice) can be used by all personal computers and most computer terminals. The service offers shopping and banking at home, electronic mail, realtime communications, current and historical stock market and commodities information, news and weather reports, games, bulletin boards, private user groups, and remote computing capabilities. It even has an entire section devoted to Commodore users, accessed with the "CBM" command.

The CIS Starter Kit comes in a three-ring binder, with a user ID number and password for five free hours of connect time. A CIS User Guide, access instructions, glossary of computer terms, an introductory subscription to *Today* magazine, frequently asked questions and answers about CIS, and rate information are also included with the kit. Suggested retail price is $39.95. See your Commodore dealer or contact:

CompuServe Inc.
5000 Arlington Centre Blvd.
Columbus, OH
43220 614 457 8600

LCS Sleeves From Periphlex

LCS (*L*ight *C*ontrol *S*ystem) Sleeves are a unique new product from Periphlex Inc. These sleeves are manufactured from black nylon micro-monofilament fibres woven in an extremely regular square weave pattern. The resulting columnating grid provides perfectly square apertures through which light passes. The nylon fibres absorb approximately 50% of the light. The result is reduced glare by reflected light creating an atmosphere with greater visual comfort over long periods.

The sleeves slip over florescent light tubes of any size. By fitting some bulbs with the sleeves and leaving others uncovered, virtually any lighting environment can be achieved. For more contact:

Periphlex Inc.
325 Milner Avenue Suite 409
Scarborough, Ontario
M1B 2V7 416 292 3800

WATSOFT CAI Authoring Program

Another product of WATSOFT Inc. is "PETCAN" ; a CAI authoring program designed to remove the programming aspect from writing CAI lessons. PETCAN prompts the author with a series of choices and, with the answers to these choices, determines the structure of the lesson. It is a "user-friendly" program, designed so that the computer novice is not overwhelmed, and the experienced author is not bogged down with lengthy explanations and procedures.

One of PETCAN's main features is complex answer checking through word comparisons, with up to eight possible comparisons for each question presented. User answers can be compared against anticipated correct OR incorrect responses.

Branching is another feature, allowing for flexible individualized lessons. Branching can be invoked based on performance, number of attempts, a specific answer (right or wrong), and user controlled branching to allow the user forward or backward movement through the lesson.

PETCAN is available now for the 4032 and 8032. Individual copies are $85. School Boards, etc., can obtain licensing for 10 copies with 10 manuals for $485. For information:

Sarah Wright
University of Waterloo
Language Laboratory Department
Waterloo, Ontario
N2L 3G1 519 885 1211 ext 2735

Bulletin Board System Version 2.03

April 83 marked the second birthday of The PSI–WordPro Bulletin out of Mississauga Ontario. Want to set up a local Bulletin Board System? Or maybe you already have. Either way, Steve Punter's latest version of his BBS software is now available.

New features include a revised 43 page manual, expanded user file records and facilities to edit and delete user records, compress to remove inactive users, system file editing facilities, message forwarding, forward and backward message reading, summary and overview from any message, list message from any user, time function, and private messages are now strictly private.

As well, new versions of both the IEEE and RS232 versions of Steve's terminal programs are included on the disk. These are also available on the TPUG Communication Disk #2 for any user that wishes a copy. Any ASCII terminal program can be used to connect with the BBS. Likewise, Steve's software can be used to access any remote ASCII installation, but only this program can be used to Up/Download programs from the BBS. Documentation for the terminal software is contained within the BBS manual, and dealers are invited to copy this section for their customers.

The BBS package is $200. Updates of the BBS software are no longer being handled by Commodore. Anyone wishing a copy of version 2.03 (or later releases as they become available) should first find their proof of purchase and serial number, and contact:

Steve Punter
1343 Tynburn Cres.
Mississauga, Ontario
L4X 1P6

Law Office Accounting Software

BPI Microsystems in consultation with the Law Society of Upper Canada have developed a law office accounting package that contains a trust accounting system, general accounting system, work in progress system, billing, payroll, accounts receivable/payable, time docketing, and will handle 3000 clients with over 5000 posting entries per month.

The package features ease–of–use, minimal operator training, owner and operator manuals with step–by–step and set–up instructions, and password protection to prevent unauthorized records access. The operator is only required for data entry and the computer will automatically perform many of the accounting and reporting functions without supervision. For more info, contact:

Doug Marks
BPI Microsystems Ltd.
705 Progress Drive Unit 17
Scarborough, Ontario
416 431 3200

Job Costing System

The Job Costing System is designed to aid in managing the cost of materials and labour, and controlling costs as they change. Features include (based on 8050 disk drive):

- tracking *any* type of job
- unlimited number of jobs (using multiple disks)
- up to 100 work categories per disk
- budget to actual report
- paid to accrued report
- numeric or alphanumeric item listing
- extensive error checking
- minimal data entry
- control totals and transaction listing
- up to 10 jobs per disk with 500 budgeted items per job

Available from all Commodore dealers, or contact Doug Marks at BPI (see previous item).

Ontario Medical Package

An efficient, powerful medical package that can handle virtually any Ontario practice with physicians in or out of OHIP.

The system can accommodate over 4000 patient records and maintain all pertinent patient data including OHIP number, status, and balance. The program supports over 100 codes for three fee schedules (OHIP, WCB, and OMA), handles basic and time units for assisting and anaesthesia, and handles premium charges. OHIP billing is completely automated, prints J2 and J8 lists, exception reports, and all ties directly into the accounting system to provide daily summaries, monthly P&L statements, among other things.

Other features include printing daily doctor and assistant schedules, keeping medical profiles on patients with allergies, continuing medication, and other medical problems, and also details upcoming appointments and recalls, and outlines how much notice patients need if an appointment must be changed. Contact BPI for more info.

PAL For The Commodore 64

PAL (*P*ersonal *A*ssembly *L*anguage) for the C64, as well as CBM 8000 and 4000 series, is Brad Templeton's full-featured 6502 assembler. Using only 4K of memory, PAL source programs are conveniently entered just like BASIC using the standard BASIC editor and helper products like POWER can be used. PAL works entirely in memory assembling programs to 1K bytes of machine code in under 10 seconds. In addition to MOS standard assembler syntax, free format input, and fully formatted listings, PAL has special features for symbol reassignment, source file chaining for large programs, conditional assembly, saving and loading symbol tables from disk and user customization of operations, code output, listings and expression terms.

PAL also features a recursive expression evalulator with eight different operators and nine different operand formats including decimal, hex, and character. PAL produced programs can be automatically relocated anywhere in memory.

PAL itself is relocatable and versions for the 8000 and 4000 series can easily be burned into EPROMS for permanent installation. Several support programs are include that allow source files in other formats to be converted to PAL format, and an un-assembler that converts machine code files back to PAL source code. PAL comes on diskette complete with documentation. Price: $99.95 (documentation is excellent, like PAL - great buy - Ed.) For more, contact:

Pro-Line Software Ltd.
755 The Queensway East, Unit 8
Mississauga, Ontario
L4Y 4C5
416 273 6350

SpellPro: WordPro Spelling Checker

The SpellPro spelling checker verifies the spelling of letters and documents at machine language speeds. Designed by Jim Butterfield for use with Commodore 8000 series computers with WordPro (C64 version due soon), SpellPro's dictionary is easily updateable to over 80,000 words on an 8050 diskette. A custom dictionary with legal, medical, or other specialized vocabulary is created by adding words from existing WordPro with the touch of a key.

Menu-driven and user friendly, SpellPro can be learned in less than 30 minutes. By rapidly eliminating misspellings, SpellPro greatly reduces clerical time spent on corrections. (not only that, but spelling checkers allow for carefree keyboarding - Ed.) Information? Contact Pro-Line above.

MailPro

MailPro, also known as WordPro MailList, is Steve Punter's exciting new multi-featured data-base like program for Commodore 8000 series computers.

MailPro is user friendly and ideal for storing and printing mailing labels, club lists, personnel records, hobby lists, product descriptions, price lists, and so on. Files are easily user defined and updated. In fact, WordPro files can be used to automatically update MailPro files, and MailPro files can be used with WordPro to customize reports and letters.

Printed output is completely user defineable and up to 10 output formats can be stored for later use. Since records are maintained in a sorted condition, printing in any sequence is rapid with no sorting delays.

On an 8050 disk, MailPro can store up to 2040 records per file with individual records containing up to 20 fields. Once again, call Pro-Line for info.

C64-Link Relocator

The C64-Link form Richvale Telecommunications comes with a relocator program for moving the Link software (ie. the IEEE routines, BASIC 4.0 commands, etc.) into higher RAM so that the cartridge area RAM can be reclaimed. For certain programs (eg. WordPro 64), the relocator must be used to move the Link software out of the way.

It was discovered, however, that even after using the relocator, WordPro 64 was still inoperative. A new version of the relocator is now available that eliminates this problem. To get a copy, send $10 for disk or $5 for tape, or phone Richvale with your Visa/Master Card number (see inside front cover).

C64 ScratchPad, Basic Aid, Termsoft

ScratchPad, the RTC free-format data base like program, will be available for the Commodore 64 this summer. More details in a later issue. Priced under $100.

Richvale is also releasing a Terminal package for around $50 and a full fledged Basic Aid at $40.

C64 Management Accounting Software

InfoDesigns has modified their 8000 series accounting package for use on the Commodore 64. General Ledger, AR, AP, payroll, and inventory modules are all priced at $199 US each.

InfoDesigns
6905 Telegraph Road
Birmingham, MI
48010 313 540 4010

C64 Farm Accounting Package

Cyberia Inc. now offers their Cyber–Farmer package on the Commodore 64. Cyber–Farmer has been in use by many farmers in the midwest for over 2 years. Retail price is $195, available from Commodore dealers or directly from:

Cyberia Inc.
2330 Lincoln Way
Ames, Iowa
50010 515 292 7634

C64 Programs from Computermat

ComputerMat announces three new program paks for the Commodore 64.

The Arcade Pak includes Alien Invasion with 20 play levels for 1–4 players. Head–On pits you against the computer car in a maze race against time. Protect cities from destruction in Target Command. Cassette $24.95, disk $29.95.

The Game Pak includes Dragon Chase, a maze escape game, Flip–It, the computer version of a 200 year old strategy game [..could this be "GO" for the 64? – Ed.], and Deflect. Cassette $15.95

The Edu–Pak contains 4 programs. Ruler puts you in control of country for as long as you last. Micro lets aspiring young entrepreneurs try their hand at running a computer manufacturing plant. Dungeon of Mathacos is a math problems treasure hunt game, and Geography lets you match cities around the US and the world. Cassette $24.95. For more contact:

ComputerMat
PO Box 1664–PR
Lake Havasu, AZ
86403 602 855 3357

The Commander

The Commander is a 4K ROM for use with Commodore 2000, 4000, and 8000 series microcomputers with BASIC 4.0. It contains about 20 commands for enhancing BASIC and program development. All commands can be used in programs or direct mode.

Several commands allow for dynamic modification of BASIC text without destroying variable tables. These are Insert, for inserting subroutines into text, Append, for appending code, and Delete, for removing lines. Each can specify a line number following the command to continue execution during program runs.

Print Using and Image are handy printer format commands. The Commander also has an enhanced keyboard GET, an array re–dimensioning command, string input up to 255 characters, array (or matrix) input, output and manipulating utilities, and more.

The Commander comes complete with ROM, manual, and demo disk for $70.00. When ordering please specify which socket ($A000 or $9000) the ROM will live at, and your disk unit type (ie, 4040, 8050, etc). For details call:

Commander Systems Inc.
4505 Jackson Street
Hollywood, FL
33021 305 962 5183

Commodore 64 COMAL

In 1981 COMAL (*COM*mon *A*lgorithmic *L*anguage) was created in Denmark by Borge Christensen for educational purposes. Since then the language has become popular world wide. COMAL combines the simplicity of BASIC with the powerful structures of PASCAL. Features include:

if–then–else–endif
while–endwhile
repeat–until
case–when–otherwise–endcase
for–endfor (next)
loop–endloop
improved string handling and
file access

COMAL checks the syntax of your program lines as they are entered and, in several cases, is much faster than BASIC. There's even a COMAL User Group and newsletter. For more information, contact either:

COMAL Interest Group
505 Conklin Place
Madison, WI, 53703

The COMAL Catalyst (newsletter)
101 Convention Center Dr. Suite 900
Las Vegas, NV, 89109

Several versions of COMAL exist. CBM 8096 COMAL and a version for the Madison Z-RAM board can be obtained from CIG (above). Instrutek, a company in Denmark, distributes a COMAL board that fits your 8032. On power-up the machine is reconfigured to a "COMAL machine" (with BASIC available on command). The board also combines hi-res graphics with two full virtual screens.

Editor's Note

At last report COMAL 64 is up and running and should be available on disk and cartridge by this summer, but no word on exact dates or price. Rumour has it that Commodore may include it on a utility disk with each C64/1541 since COMAL *is* public domain software. I received a copy of the new COMAL Handbook ($18.95 US from CIG) which looks very well done. (by the way, COMAL can be obtained on disk from CIG for a $15 US handling fee, or send $29.95 and get the disk and the handbook) In my opinion, COMAL is the first structured language one should learn, and the first high-level language young students should be instructed in. Most universities frown on the use of GOTOs and other simplistic, non-structured formats. Once someone learns a language like BASIC, it's easy to get stuck in a rut that's potentially un-escapable. We'll be discussing the history, concept, and direction of COMAL in an upcoming issue. Watch for it!

Hardware News

Memory Expansion For The C64

No mistake. Richvale Telecommunications is developing a memory expansion board for the Commodore 64 for release sometime around July 83. The board can be fitted with various amounts of memory and an optional 68000 cpu.

With the maximum memory configuration – 512K – one can write BASIC programs that fill all 512K! A special BASIC is supplied that swaps out the resident BASIC in the 64. BASIC 4.0 commands are included with true BASIC 4.0 garbage collection!

Another application is the *RAM Disk*. The entire contents of a floppy are loaded into the expansion unit and subsequent file retrieval occurs at lightning speeds directly from RAM.

No details on price, but wait 'till at least June before swamping them with calls.

Power-Line Protection

Severe AC Power-Line spikes, surges, and hash are prevalent in many microcomputer installations. Program execution is plagued with unexplained crashes, memory loss, or other glitches. Disks, printers, and processor often interact, aggravating the problem.

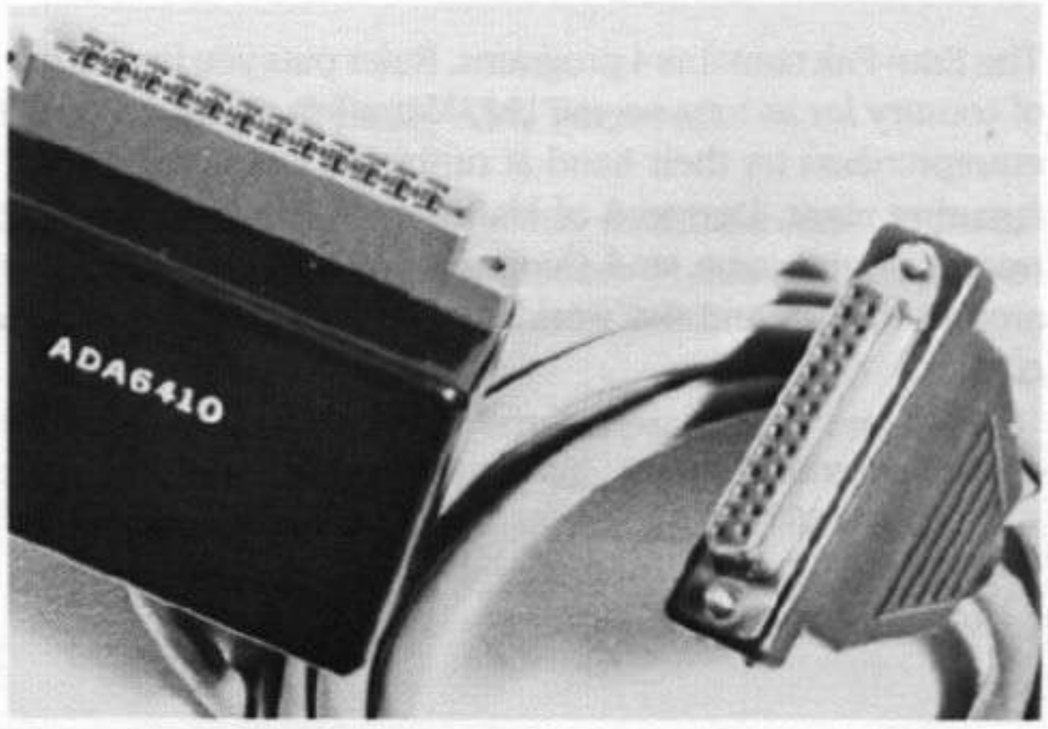
Electronic Specialists recently announced Model ISO-11 is designed to protect software from these severe electrical problems. Complementing the popular Super-Isolator line, the Model ISO-11 features two individually dual-Pi filtered AC socket banks (6 sockets total). Heavy-duty spike/surge suppression is incorporated in the design. Equipment interactions are eliminated, disruptive or damaging spikes and hash are controlled, and programs operate without interference. Price is $115.95:

Electronic Specialists
171 South Main Street
PO Box 389
Natick, MA
01760 617 655 1532

C64 To RS232 Interface Cable

Connecticut microComputer Inc. introduces the ADA 6410, an interface cable for the Commodore 64 to RS232.

The cable plugs into the RS232 Port (ie. User Port) and provides voltage conversion to drive standard RS232 printers, terminals, and mainframes.



The unit is complete with six feet of cable, and all electronics are completely enclosed. Power is received from the computer. There is no special software needed – secondary address #2 does the rest.

Connecticut microComputer
36 Del Mar Drive
Brookfield, CT
06804 203 775 4595

BackPack Battery Backup

BackPack, a battery backup system for PET/CBMs and floppy disk drives, will prevent loss of data in the event of a power failure. BackPack supplies a minimum of 15 minutes reserve power to 32K of memory, the video screen, and tape drive. The unit is easily installed right inside the cabinet. Recharging occurs during normal equipment operation and has an integral on/off switch.

From complete discharge, BackPack takes 16 hours to reach full charge, and has a life expectancy of three to five years. Models are available for all dual drive disks, 4000/8000 series computers, and the C2N cassette deck. For further info:

Jeff Butler
Electronic Technology Corp.
PO Box G
Old NC 42
Apex, N. Carolina
27502 919 362 4200

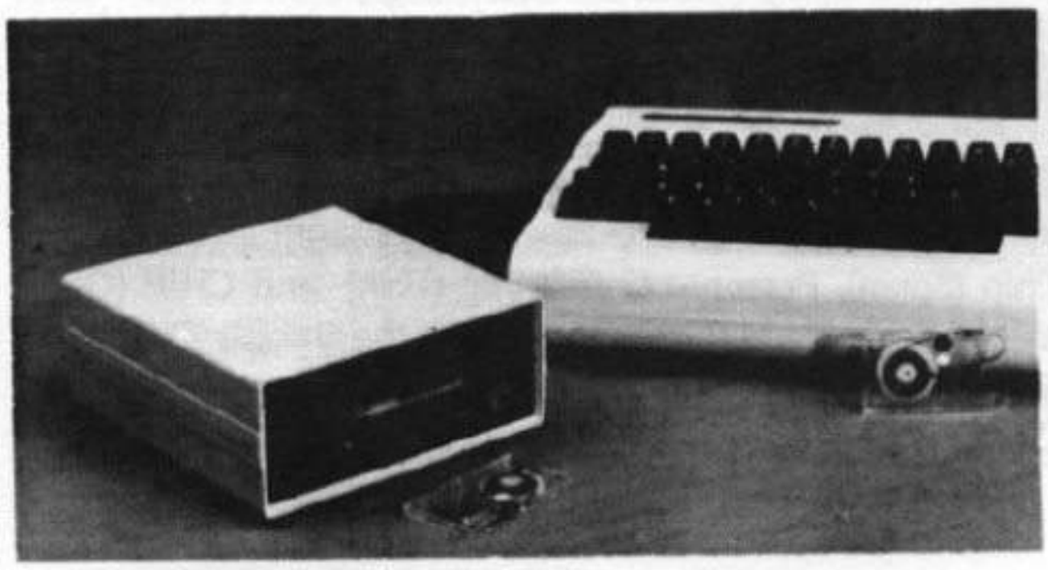
VIC 20/C64 Stringy Floppy

Exatron has announced the ESF–20/64 String Floppy drive for use with the VIC 20 or Commodore 64. The unit has a serial bus and connects to the computer just as a 1541 disk drive would. The difference of course is the storage media; a miniature tape cartridge containing an endless loop of magnetic tape.

Features include reliability, speed, compact size and compatibility with software (ie. BASIC text AND BASIC ROMs). The drive operates at 5 inches of tape per second and has a capacity of up to 64K per tape cartridge. Previously stored data is easily transferable to the cartridges using simple BASIC commands.

Retail price is about $199. Additional tapes are about $3 each. For more:

Exatron
181 Commercial Street
Sunnyvale, CA
94086 408 737 7111
outside CA 1 800 538 8559



IEEE–488 Data Aquisition Module

Connecticut microComputer announces a new 64 channel digital line input module for use with IEEE–488 bus compatible equipment. It is the first product in the BUSSter series of I/O modules.

The BUSSter A64 Digital Input Module accepts commands from any host computer through its IEEE port, reads and store data from up to 64 digital TTL level lines and then sends this information back to the computer. A BUSSter module economically increases the interfacing capability while reducing its workload.

The A64 is easily programmed through BASIC commands from the controlling computer. The built–in timer and buffer allow data sampling and collection to occur while the host is occupied with other tasks.

The BUSSter A64 sells for $495.00 including case and power supply. Further info available from:

Joanne Akin
Connecticut microComputer
36 Del Mar Drive
Brookfield, CT
06804 203 775 4595

Commodore News

VIC 20 Voted Computer of the Year

Vancouver – An international competition run by seven personal computer mags has recently voted the VIC 20 "Home Computer of the Year". Judging was held in London, England.

Competition sponsors were Databus (Holland), Microsystems (France), Bit (Italy), Practical Computing (England), Chip (Spain), Personal Computing (USA), and CHIP (Germany). Second and third place went to the Sinclair ZX81 and Spectrum. Runners up were the Atari 400 and the TRS Colour.

Commodore and Zilog Announce Agreement

Irving Gould, Chairman of the Board of Commodore International announced February 2nd that CBM had signed a definitive agreement with Zilog Inc., an affiliate of the Exxon Corporation.

"The agreement refers to an exchange of technology between Commodore and Zilog, with Commodore licensed to manufacture Zilog's Z8000 16–bit microprocessor and related family of peripheral support circuits for use in Commodore microcomputer systems, while Commodore will supply Zilog with mask sets and manufacturing rights to selected custom circuits used in current and future Commodore microcomputer systems" , said Gould.

"Commodore's financial business commitment to Zilog is significant and involves a similar commitment by Zilog that could utilize a substantial portion of Zilog's production facilities devoted to the manufacture of large scale integrated circuits for exclusive use in Commodore microcomputer systems", he said.

This agreement is intended to give CBM a substantial increase in semiconductor manufacturing capacity toward meeting its large scale integrated circuit needs, while reducing the time and investment required if Commodore were to expand its own semiconductor manufacturing capacity. "Notwithstanding this," Gould added, "our own semiconductor facilities' capacity is in the process of being increased 100% as part of a program to be completed over the next twelve months."

Free Program Pack with 1541 Disk Drives

1541 Disk Drive purchasers will now receive a Commodore 64 program pack at no charge. You get a manual with short descriptions of over 30 public domain programs including some games, sound/graphics demos, and utilities.

C64 Programmers Reference Guide

The Commodore 64 P.R.G. is now available from Commodore dealers. If you plan to do anything serious and haven't already got one, GET ONE! About 500 pages and great value for money.

Convert PET Programs to C64 Programs

"Converting PET Programs for the Commodore 64" is a manual detailing potential software incompatibility problems between the two machines. Most programs are portable from the PET to the 64, but others will require major surgery. This of course refers to PEEK, POKE and WAIT statements, and SYS & USR commands which invariably indicate the presence of machine code. Topics covered include Upper/Lower case and graphics displays, output to the screen, clearing the keyboard buffer, disabling RUN/STOP, and may more. See your dealer for this and the 64 PRG.

New Commodore Colour Monitor

Commodore 1701 Colour Monitors should be available from dealers now at $539.95 Cdn. The unit has a 14" screen, 7 different picture controls, and offers outstanding resolution, superb colour clarity, and a built–in speaker. It also has separate luminance and chroma inputs which make it a perfect companion for the 64. (I had a chance to see one at Commodore and even the reds and yellows come through clear – excellent value $539 Cdn. – Ed.)

C64 Keyboard Update

Paul Higginbottom of Commodore Canada is nearing completion of his keyboard synthesizer package for the C64. He says, "The keyboard unit will now have 49 keys (4 octaves), as opposed to 37 as originally planned. Also, the keyboard itself will house an 8K "keyboard kernal" ROM, containing routines such as set a voice, clear a voice, set frequency, etc. The ROM will map into the normal cartridge slot area on power–up and offer a standard jump table for those who

want to write their own software without a lot of *re-inventing the wheel*. Full documentation of the ROM routines will be supplied."

[I've been following progress on this project since Paul began work on it. For about $1800 Cdn, which includes a C64, the keyboard unit and cartridge (about $200 Cdn!), a monitor, and a disk drive, you can have a stage-quality synthesizer that's virtually untouchable, even by synthesizers costing twice as much. I plan to get one and I can't even play the sucker! – Ed.]

New Commodore Printers

The 6400, 4023, and 1526 are Commodore's latest additions to their printer line.

The 6400 is a letter quality printer (daisy wheel) manufactured by TEC. This will replace Commodore's previous letter quality printer made by Diablo, the 8300P. The 6400 has an IEEE interface and is fully compatible with PET ASCII. Features include bi-directional print, paper out switch, set top, set left, pitch control, underscore, bold and shadow print, reverse scrolling, and automatic hammer strike intensity (determined by the surface area of the character). Price around $2800.

The 4023 replaces the 4022. Both are virtually the same printer, except this time Commodore's done it right. The 4023 has an 8 by 8 dot matrix that will offer accurate representations of every graphic character! Printing is bi-directional – about 60 cps – with friction or tractor feed.

The 1526 is exactly the same printer, but with a serial interface compatible with the VIC 20 and Commodore 64. See your local dealer for details on price and availability.

Commodore 64 LOGO and PILOT

Commodore has contracted with the original authors of this favorite graphics language to put it on the Commodore 64. The adaption will be complete and similar to the Apple LOGO, with extensions that take advantage of the 64's sprite graphics capabilities.

Key features: graphics definition and movement, text display, multi-level graphic character display, complete screen editing, and comprehensive documentation.

PILOT contains all the "Common PILOT" standard features, plus extensions. Most PILOT programs already available for other machines should run on this version with little or no modification. Interface instructions are also included for connection to Video disk/tape players.

Both languages are available from Commodore dealers for somewhere in the neighborhood of $150 Cdn.

Commodore Software News

IBIS CAI Program Developer

IBIS – Interactive Basic Instructional Structure – is a newly released program that writes educational software in BASIC for Commodore computers [not unlike PETCAN see earlier note. Ed.]

IBIS can reduce program development time by 10 to 20 times because it allows people who may have no knowledge of BASIC to produce CAI lessons that run first time. One rather interesting feature is that IBIS actually creates a BASIC program that can be modified using the BASIC Editor.

IBIS runs on any Commodore computer with 32K and BASIC 4.0. It comes with a cassette port protection key and full documentation. A Commodore 64 version is also being considered.

Commodore Computer Educator Systems

The Computer Educator Systems were designed to assist students at various grade levels by providing a fun and stimulating learning series to help them become more involved in education, however, many adults would also learn from these programs.

Four main categories are covered: Science, Language, Social Science, and Mathematics. The Science series is aimed at grades 3 to 10 with sections on elementary life sciences, and introductory physics and chemistry. The Language series deals with grammar, spelling, and vocabulary skills in English and French. The Social Science series covers history (N. American, ancient, and middle ages), and world geography with emphasis on Canada and N. America. The Math series includes levels from grades 1 through 11; simple arithmetic through algebra and geometry.

Each section of each series come in packs of six tapes with manuals for $69.95 per pack (suggested retail). See your dealer for details.

Commodore 64 "Easy Software"

The "Easy Software Series" available from CBM on previous product lines is now available for the 64. Among the programs in the series are:

Easyscript64 (wordprocessor)
EasyCalc64 (spreadsheet)
EasyPlot64 (graphics plotter)
EasyFile64 (data base)
EasyMail64 (mail list)
EasySchedule64
EasyFinance64

All programs come on 5 ¼ inch floppies with documentation for each. For more information, contact any Commodore dealer.

SkuttleBits

P128 Changed to C128

For no *apparent* reason, Commodore has decided to change the name of the (as yet publicly unreleased) P128 machine to the "C128" . This is the low profile machine with numeric pad, 128K RAM, 6509 processor (which is NOT a 6809 stereotype for those that noticed the part# similarity), and no screen. The "B" and "BX" model numbers remain unchanged.

Some industry concern has been expressed about the "programability" of the "C" and "B/BX" models in machine language. Both computers allow for scads of BASIC through an elaborate system of bank-switching. However, machine language programs larger than a single bank might have trouble coping with themselves. (a rather strange statement to make, I agree) At last report, Commodore will be including ROM code that will eliminate any potential problem, but here's the details that prompted the doubt.

It seems that 6509 architecture didn't allow for calling and returning from subroutines that lived in other banks besides the bank containing the calling instruction. Think about it. If bank F (for example) contains a call to a subroutine that lives in bank 1, bank 1 must first be switched in. Now the JSR instruction in bank F has been switched out. When the code in bank 1 comes to an RTS, execution must somehow get back to bank F, information which is not normally stacked by the 6809. The kernal routines that handle this situation are examined in detail by Jim Butterfield next issue.

One last note. . . the "C" and "B" series machines will include a machine language monitor (which tends to suggest right there that the previously mentioned problem has indeed been eliminated). To make room for the MLM, the tape I/O routines had to be removed, but are provided for through an internal vector. Simply stated, the machines will come with a cassette deck port that is non-functional until extra software is loaded. This utility will be included with the unit, however most users of either machine will probably operate with disk drives.

SX-100 Now SX-64

The SX-100 will now be known as the SX-64. Recall, the SX-100 was the Commodore 64, a 1541 disk, and a 6 inch colour monitor. . . all in one box like the Osbourne 1. Price is around $1500 but still no release date available.

New Approved Products Program

Commodore is about to complete an approved products policy that will govern ads placed in CBM publications (ie. Commodore Magazine and Power/Play). Specifically, ads submitted to either magazine won't be accepted unless the product is officially approved. Here's an excerpt from a preliminary announcement:

> ". . .We aren't enforcing this particular policy [Feb 11/83] until the details of the program have been worked out, but we want to give you an idea of what is coming.
>
> The program will serve both our readers and our advertisers. Since many of our readers are new to the world of computing, and since the acceptance of ads in a Commodore-sponsored publication is a *de facto* endorsement, we want to be sure of each product.
>
> More important, this effort represents a genuine commitment on Commodore's part to third-party vendors. Among the details being discussed for the program are financial and design assistance for advertising the products (in any magazine, not just our own), technical support and product testing, special shows open to approved products only, and production of an approved products catalog for inclusion in Commodore's product packaging."

From the sound of things, Commodore has taken one more step towards getting their act together – a good sign. But I can't help chuckling over this additional bit of info that came in with the same lot. . .

"Re: Using Commodore trademarks in your ads

In order to protect the integrity of our company's trademarks, we enforce the following trademark regulations for all advertising in our user magazines:

1. All Commodore products mentioned by name must bear either a ™ or an ® immediately following the first mention of the name. The present trademark status of our products is:

PET® CBM® VIC 20™
SuperPET™ Commodore 64™

2. Product names must be spelled correctly. Please note that the VIC 20 has a space but no hyphen and the Commodore 64 is called only that – not CBM 64 or C–64 or any such variation.

Any ad that does not meet these requirements cannot be run in our magazines, so please check over your ads carefully before sending them in."

Point #2 is the bit I'm referring to. I agree with this policy as far as ad content appearance is concerned – it provides for some form of continuity. But I always thought that nicknames were a sign of popularity, something a manufacturer would naturally welcome. For example, I'm sure that "Chevy" was not invented by GM. Further, several aftermarket vendors have already incorporated variations into their trade names (eg. C64–Link). I'm sure Commodores' intent was not to cause any discomfort, and besides, point #1 technically exempts this situation because it does not actually mention a pure Commodore product by name. No doubt, good judgement will be employed before policy details are released.

I also hope Commodore won't object to authors using product name variations in published articles (the announcement suggests they won't). Referring to "the 20" and "the 64" is an easy way of breaking monotony. From now on "The T" will post Commodore trademark status on our own policy page. But typesetting those superscripted TMs on every occurence could get to be a royal pain in the A register. I hope they won't mind if I omit them.

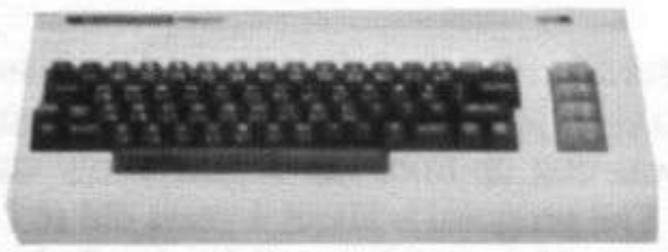
I also hope they won't mind this next zinger, because let's face it, if we can't laugh at ourselves, who can we laugh at. The following clippings were taken from recent Commodore information releases.

COMMODORE SENDS UNIQUE CHRISTMAS GREETINGS TO C= 64 DEALERS

Some of you probably think I should have put all this in an editorial. But when you're a blabbermouth like me, editorial space becomes somewhat precious.

It looks like we're going to have two sets of standard product names; one for advertisers and another for general logomachy and confabulation. Here's how I affectionately refer to my favorite microcomputers. . .

The Max



The 20

The 64

I hope this doesn't catch up with me. . . trouble has a way of following me around. Besides, it's all in good fun.

And now, back to our program...

Bits and Pieces

Kaleidoscope

This program was dug up from the depths of my cassette tape collection. The program is about 4 years old so I don't know who wrote it, although I'd be delighted to find out.

```
100 print "■";:c=0
110 for j=0 to 7 : read ch(j) : next
120 data 160, 127, 102, 64, 91, 93, 58, 32
130 sc=32768 : cols=80 : lines=25
140 mx=int(co/2-1) : lc=li/co : ck=co-0.0001
150 for h=3 to 50
160 for i=1 to mx
170 for j=0 to mx
180 k=i+j
190 c=ch((j*3/(i+3)+i*h/12) and 7)
200 s1=sc+co*int(lc*i)
210 s2=sc+co*int(lc*k)
220 s3=sc+co*int(lc*(ck-i))
230 s4=sc+co*int(lc*(ck-k))
240 poke i+s2, c : poke co-i+s2, c
250 poke i+s4, c : poke co-i+s4, c
260 poke k+s3, c : poke co-k+s3, c
270 poke k+s1, c : poke co-k+s1, c
280 next j, i, h
290 goto 150
```

The program was modified to work on all Commodore machines so it doesn't make use of colour on the VIC and 64. Before RUNning, change the variables in line 130 to suit. SC is the screen start address, the other two are rather obvious.

To get colour wouldn't be hard though. All you would need is another variable, say CT for Colour Table, set equal to it's start address. Then just copy lines 200 to 270 into all the "in between" line numbers (ie. 205 – 275) and substitute CT for SC, C to CL, and C1 to C4 for S1 to S4. The variable CL is a colour that would be generated randomly, or using another cryptic statement like line 190. Right now line 190 chooses a character from the CH array which is set up with the screen poke values of 8 graphics. Change these if you wish, but likewise, another array (eg. CL(0–7)) could be set up to contain the poke values of some screen colours from which a new line 195 would select. Of course if line 195 were stereotyped from line 190, each character would get the same colour in all occurences. Try changing some I's and J's around.

VIC 20 users and Commodore 64 users with Kernal 2 will need to add this line:

```
C64 : 105 poke 53281, 13
V20 : 105 poke 36879, peek(36879) and 15 or 208
```

This changes the background colour so that the pokes to the screen will show up. With the 20 (and 64 Kernal 2), a clear screen writes the whole colour nybble table with the background colour value. Thus a poke to the screen only puts a character in screen memory that's the same colour as the background, making it "invisible". This will be averted if simultaneous colour selection is added, however, changing the background colour dynamically might also be interesting.

As is, the program's about as fast as it's gonna be in BASIC. If you have a compiler and you've got nothing better to do. . . Finally, any dazzling new versions would be most welcome in a future issue!

4.0 Disk Append

The program to follow is fairly self explanatory.

```
10 print "basic 4.0 disk append
20 print "this routine will allow a subroutine
30 print "saved as a program file on disk to be
40 print "appended to a program in memory." : print
50 print "the subroutine must begin with a line
60 print "number greater than the last line of
70 print "basic text in memory          " : print
80 print "activate with : ";
90 read ad : rem replace with ad=??? for permanent placement
100 for j=0 to 56 : read x : poke ad+j, x : next
110 print "sys" ad; chr$(34) "file name" chr$(34) ",8"
120 rem routine is fully relocatable
130 rem first data element is start address (ad)
140 rem remove 1000 if ad is set within program
1000 data 634
1010 data 169,   0, 133, 157,  32, 125, 244, 169
1020 data  96, 133, 211, 164, 209, 208,   3,  76
1030 data   0, 191,  32,  73, 244,  32, 165, 244
1040 data  32, 210, 240, 165, 211,  32,  67, 241
1050 data  32, 192, 241,  32, 192, 241,  56, 165
1060 data  42, 233,   2, 133, 251, 165,  43, 233
1070 data   0, 133, 252,  32, 140, 243,  76,  28
1080 data 244
```

Lately I've had quite a bit of use for it. For example when you want to renumber only part of a program and don't have a selective renumber utility. First you renumber the part you want renumbered, then you bring in the rest with DiskAppend.

The version above works on BASIC 4.0 only, however if there's enough demand we'll re-cut it for the others. After running it once, I tend to save a direct load copy from the MLM. This way it can be loaded back using the monitor without disturbing the contents of memory.

Crash Your Commodore 64!

A dastardly perpetration indeed for such a fine upstanding computer. But let's do it anyways (Nyah ah ah). While looking for the problem with the C64 POP SYS published last issue (see the Transbloopors section this issue for more) I stumbled across a most interesting crash, however it seems to only work on 64s with the racing stripes:

```
poke 783, 8 : sys 42622
```

Now hit some number keys. That's it. . . keep going. Neat eh? (Neat uh? for U.S. readers). And why is the text on the screen (ie. from above) not disturbed by all the vertizontal scrolling? Hmm. Eventually it locks up completely but no harm done. Just power down, up, and you're back to normal (your machine that is). Any more out there?

C64 TV Colour Adjust

Don Lekei of Vancouver, B.C., has this useful note for those not satisfied with the colour output of their C64 to a television set.

First power down your 64. Open the casing and on the PC board, around the general vicinity of the Return key, you'll find a metal "box" (you can't miss it). Lift the lid off this box, being careful not to wipe off that white gunk that's on the bottom of the lid and the top of that 40 pin chip. This stuff acts as a heat transfer from the chip to the lid.

Inside the box are two white nylon adjustment pots. The one on the left is the Chroma output adjust, and the one on the right is the Clock rate adjust. Turn your 64 back on and, of course, connect it to your TV. Before changing these pots, take a black felt pen and mark each one so you can return to the original positions should you get carried away. Also, you'll need some stuff on the screen to make the adjustment by. Set a black background with POKE 53281,0 and type some lines of jibberish on the screen using the colours available from the keyboard (ie. CTRL - WHT, RED, CYN, etc.). Once that's set up you're ready for the important part.

Using a small screwdriver (preferably plastic in case it gets dropped), turn the Chroma level down (counter-clockwise) until your TV just looses the colour signal. Ideally it will "flicker" between colour and no colour. Now adjust the

Clock pot so that colour is regained and stable. This will be either a clockwise or counter-clockwise turn, but all it should need it just a "tweek". Look good? Re-assemble the machine, making sure the metal lid goes on the right way, and you're done.

Of course you may not get it looking any better than it was. If so, you can always re-set both adjustments to their original positions. If your machine is still under warranty, you may want to take this note and your 64 down to your dealer and have them do it.

CRless CMD

How many of you are making sequential listings of programs to your disk drive? If you are, you probably do something like:

```
open 8, 8, 8, "0:prog file,s,w"
cmd 8 : list
print#8 : close 8
```

This puts a listing to the disk just as if it were to the printer. You might use this file at some later time with a terminal program, etc. But have you noticed an extra carriage return always seems to creep in at the beginning of the file? If this bothers you, read on. If not, forget I mentioned it.

The CMD command works somewhat like PRINT#. Without punctuation a CR gets sent afterwards. In future, try replacing the CMD 8 with **CMD 8,;** This suppresses that extra preceding CR. And just a reminder, choosing a file number 128 or greater causes a Line Feed to be sent after each Carriage Return.

For some reason the comma cannot be omitted. Does this suggest there may be other combinations?

Sunny Side Up!

It seems almost criminal what they're charging for diskettes these days. The material can't cost more than about a quarter, and they probably make about a thousand every minute. So what do we do? We cleverly hack out a notch on the opposite side of the jacket, flip them over, and presto! One double sided diskette.

But consider this. A diskette always spins in one direction, even double sided disks, which are not intended to be flipped but rather for use with double head drive units. When dirt and dust particles manage to land on the disk surface, the drive spins it into the inner lining of the jacket. Usually it collects there and doesn't bother us too much. If the disk is flipped over, rotation will now be in the opposite direction. All that crud will now be released from the lining on both sides of your disk. YUK!

I hope I'm not sounding like a diskette salesman. I suppose we just have to put up with prices until enough competition brings them down. . . kind of a "fact of datalife" (ooh, poor).

This tip and others will be published in an all encompassing general care article planned for a later issue. We're still collecting more so if you have any helpful advice for others, send it in and we'll be sure to include it.

Waste Space

The following is for those who like to define your variables at the beginning of a program. If you're the type that lets all your variables be defined as execution sees them, you could be in for some peculiar results.

When BASIC "sees" a variable, the variable name and information related to it gets stuffed into the simple variables table that sits immediately above your program text. Each new variable seen causes the table to grow by 7 bytes, which cannot be reclaimed short of doing a CLR. But only floating point variables make use of all 7 bytes. String variables use only the first 5, the last 2 are wasted, and integer variables use only the first 4. MicroSoft did it this way to reduce ROM code and also to make searching through the table a little quicker.

I don't suggest you actually try to use these bytes unless you're seriously strapped for memory and another 7 bytes to store some value is absolutely unaffordable. This was meant more as a refresher on the simple variables table than anything else. However if you do need these bytes, here's how to go about it.

First you MUST define some variables at the start of your listing that you will be using later on. These variables must be either string or integer type, but depending on how many you choose to pre-define will decide how much waste space you'll have to play with later. If you're using J.B.'s String Thing, stop right here. Jim uses the last two bytes of the first variable definition, and assumes these will start with 0. Watch out for this with some other utilities too.

Moving Strings

To demonstrate the above, I borrowed part of a "function input" idea I was tossing around. . . more on that in a minute.

Line 100 defines some variables. As you can see, we'll only be able to use the waste space from A$ and B%; the third table entry, C, must not be disturbed!

To save space, the poke/peek address is calculated in a function. FN AD takes the address stored in the Start of Variables pointer and adds Y to it (equivalent of "indirect indexed" in machine code).

```
100 a$="" : b%=0 : c=3.3
110 def fn ad(y)=y+peek(42)+peek(43)*256

500 input "some function";fu$
510 a$=fu$
520 gosub 1000
530 rem go to plot routine, etc.

1000 poke fnad(5), peek(48)          :rem save bottom of
1010 poke fnad(6), peek(49)          :rem strings pointer
1020 poke 48, peek(fnad(2))-1        :rem move pointer to
1030 poke 49, 2                      :rem basic input bufr
1040 poke fnad(12), peek(47)         :rem save Top of Arrys
1050 poke 47, 1                      :rem move pointer below
1060 a$=a$                           :rem $0200 & rebuild a$
1070 poke 2*256 + peek(fnad(2)), 0 :rem 0 marks bufr end
1080 poke 47, peek(fnad(12))         :rem restore Top of Arys
1090 poke 48, peek(fnad(5))          :rem restore string
1100 poke 49, peek(fnad(6))          :rem bot pointer
1110 a$=""                           :rem null a$
1120 rem sys to tokenize routine, and/or eval expression
1130 return
```

Line 500 inputs FU$ (sorry about my choice of variables but FN$ would cause ?Syntax error, and you have to admit, it probably wouldn't be used elsewhere). FU$ is transferred to A$ because A$ must be nulled later on.

With a Y argument of 5, the result of FN AD is the effective address at which we store the Bottom of Strings pointer. To store info in B% waste space, simply pass a 12 (5+7, and 13, 6+7) to FN AD.

Now the String Move part. Lines 1020 and 1030 alter the Bottom of Strings pointer to point at the BASIC input buffer. The buffer starts at $0200 (dec 512), but the pointer is set to $0200 plus the length of A$, which we get from PEEK FN AD(2) (note: this better not be >80). Before A$ is rebuilt, the Top of Arrays Pointer must be set lower than the Bottom of Strings pointer so that garbage collection is not invoked which would ruin everything. Line 1060 merely causes A$ to be transferred to the BASIC input buffer, and line 1070 marks the end of valid buffer contents with a 0.

After mending all the assaulted pointers, again using FN AD to access values stored in waste space, A$ is set to null so that garbage collection doesn't clobber anything important down in lower memory. Continuing, the program might continue to use the contents of the input buffer. Specifically, the string could be tokenized and evaluated for use with a plotter subroutine, trigonometry program, etc.

I haven't actually tested this, but the idea was to demonstrate using waste space. For one thing, the contents of the BASIC input buffer may require a conversion to screen codes before tokenization is called. For another, the buffer itself may get clobbered by subsequent BASIC operations (ie. POKE, FN, SYS, etc.). If someone decides to tackle the "function input" utility, it may require machine code for everything past the input statement. However, I'd be most interested in ANY results!

Butterware

The next three programs are from who else but Jim Butterfield. The first is String Thing 64, the second is Tapemaker for BASIC 4.0, and the third is Universal Disk Change.

String Thing 64 is the Commodore 64 version of Universal String Thing, published in Transactor 01, Volume 4. This utility will input strings from the disk up to 255 characters in length, terminating only on carriage return. . . handy when commas and colons in files are interfering with input. Remember, the variable that receives the input must be the first one defined in the simple variables table. Also, the file number used to open the input file must be number 1. (ie. OPEN 1, 8, etc.)

String Thing has other interesting applications. The test for CR can be changed to virtually any character (see Catstrapolator this issue). Input is also controlled by the length of the buffer string (A$ below). If lines 110 and 120 are removed, input will be throttled to 15 characters maximum (len(a$)). Location 142 stores the number of characters received from the most recent input (note lines 430, 440).

I can't be sure, but a quick look at the machine language part indicates that it will work on the VIC 20 without modification.

```
50 rem ********************************
60 rem **          string thing 64     **
70 rem **          jim butterfield     **
80 rem ********************************
90 rem input string must be first variable
100 a$ = " abcdefghijklmnopq "
110 a$ = a$ + a$ + a$ + a$ + a$
120 a$ = a$ + a$ + a$
130 rem above sets string for max (255)
200 data 160,   2, 177,  45, 153, 137,   0, 200
210 data 192,   6, 208, 246, 162,   1,  32, 198
220 data 255,  32, 228, 255, 201,  13, 240,  15
230 data 164, 142, 145, 140, 200, 132, 142, 196
240 data 139, 240,   4, 165, 144, 240, 234,  76
250 data 204, 255
260 for j = 896 to 937 : read x : ch = ch + x
270 poke j, x : next j
280 if ch<>6120 then stop 300 stop
400 open 1, 8, 2, " file "
410 rem next sys same as 'input#1,a$'
420 sys 896
425 rem l = size of input (could be 0)
430 l = peek(142)
440 print left$(a$,l)
450 if st = 0 goto 420
460 close 1
```

Tapemaker

Tapemaker will accurately make a tape from any PRG file on disk. Now you might say, ". .but all I have to do is DLOAD the program and SAVE it on tape for the same results". Not so! Not all programs load to the start of BASIC text space. Some load to the cassette buffer and others load above the start of BASIC. Tapemaker will handle these correctly.

And there's more! Tapemaker will also make tapes of PRG files that start above address $8000. The tape routines in ROM don't allow this, even if you use the machine language monitor. Now you ask, ". .what would I possibly want to save on tape that lies above hex 8000 ?".

Tapemaker is EXTREMELY position dependent. One too many or too few spaces and whammo! So in order to publish it and eliminate any potential for entry error, we've taken the entire program and put it in DATA statements. Since you need a disk drive to use Tapemaker, the two listings that follow actually generate a new program on your disk. In fact, don't waste time making them too pretty. . . once the listing has been entered and RUN, you can discard it. . . you won't need it again.

```
100 for j = 1 to 348 : read x : ch = ch + x : next
110 if ch<>32053  then stop
120 open 8, 8, 8, " @0:tapemaker 4.0,p,w
130 restore : for j = 1 to 348 : read x
140 print#8, chr$(x); : next : close 8 : end
150 data   1,   4
160 data  44,   4, 100,   0, 153,  32,  34,  84
170 data  65,  80,  69,  77,  65,  75,  69,  82
180 data  32,  32,  32,  32,  32,  32,  32,  32
190 data  32,  32,  74,  73,  77,  32,  66,  85
200 data  84,  84,  69,  82,  70,  73,  69,  76
210 data  68,  34,   0,  64,   4, 110,   0, 160
220 data  49,  58, 159,  32,  49,  44,  56,  44
230 data  49,  53,  44,  34,  73,  34,   0,  98
240 data   4, 120,   0, 160,  50,  58, 133,  34
250 data  78,  65,  77,  69,  32,  79,  70,  32
260 data  70,  73,  76,  69,  32,  79,  78,  32
270 data  68,  73,  83,  75,  34,  59,  78,  36
280 data   0, 117,   4, 130,   0, 159,  50,  44
290 data  56,  44,  51,  44,  78,  36, 170,  34
300 data  44,  80,  34,   0, 144,   4, 140,   0
310 data 132,  49,  44,  69,  44,  69,  36,  58
320 data 139,  69, 179, 177,  48, 167, 153,  69
330 data  36,  58, 137,  49,  49,  48,   0, 166
340 data   4, 150,   0, 133,  32,  34,  80,  69
350 data  84,  32,  79,  82,  32,  86,  73,  67
360 data  34,  59,  86,  36,   0, 192,   4, 160
370 data   0, 151,  56,  50,  54,  44,  49,  58
380 data 139, 198,  40,  86,  36,  41, 178,  56
390 data  48, 137,  32,  49,  56,  48,   0, 219
400 data   4, 170,   0, 151,  56,  50,  54,  44
410 data  51,  58, 139, 198,  40,  86,  36,  41
420 data 179, 177,  56,  54, 137,  32,  49,  53
430 data  48,   0, 229,   4, 180,   0, 158,  49
440 data  50,  54,  53,   0, 239,   4, 190,   0
450 data 160,  50,  58, 160,  49,   0,   0,   0
460 data 162,   2,  32, 198, 255,  32, 228, 255
470 data 133, 251, 133, 201,  32, 228, 255, 133
480 data 252, 133, 202, 169,  98, 162,   5, 133
490 data  90, 134,  91,  32, 228, 255, 160,   0
500 data 145,  90, 230,  90, 208,   2, 230,  91
510 data 230, 201, 208,   2, 230, 202, 165, 150
520 data 240, 233,  32, 204, 255, 160,   0, 132
530 data 150, 200, 132, 212, 200, 177,  42, 133
540 data 209, 200, 177,  42, 133, 218, 200, 177
550 data  42, 133, 219,  32, 149, 246,  32, 140
560 data 248, 173,  58,   3,  32,  25, 246, 169
570 data  98, 162,   5, 133, 251, 134, 252, 165
580 data  90, 133, 201, 165,  91, 133, 202,  76
590 data 206, 248
```

Now LOAD Tapemaker using the appropriate file name. A LIST will show you the BASIC part but beyond this is a

whole mess of machine language. DO NOT make changes. Altering the BASIC code will shift the machine language and the SYS command will probably crash your computer. Changing the SYS command won't help either 'cause the machine code is position dependent too!

When Tapemaker is run, it will prompt for the filename of the program you wish to make a tape of, and then it asks "pet or vic?". This refers to the name you just entered, not the machine it will be loaded into. That is, if you're making a tape of a PET program, enter 'p'. Enter 'v' if the filename you entered belongs to a VIC or 64 program, OR if you want the tape to be a "direct load" file such as with machine language programs. Then simply follow the instructions to come.

Universal Disk Change

Disk Change allows you to temporarily change the device number of any Commodore disk unit; 2040, 4040, 2031, 8050, 8250, 1540, and 1541.

```
100 data 12, 50, 119, 0
110 input "old device number";do
120 if do<8 or do>15 then 110
150 input "new device number";dn
160 if dn<8 or dn>15 then 150
200 open 15, do, 15 :rem command channel
210 a$=chr$(do+32): b$=chr$(do+64)
220 read a : if a=0 then print "disk not recognized!" :
    goto310
230 print#15, "m-r"chr$(a)chr$(0): get#15, x$ : if x$<>a$
    goto 220
240 print#15, "m-r"chr$(a+1)chr$(0) : get#15, x$ : if
    x$<>b$ goto 220
300 print#15, "m-w"chr$(a)chr$(0)chr$(2)chr$(dn+32)ch-
    r$(dn+64)
310 close 15
```

To get your unit back to the original device number (usually 8) simply power down. Or you can send this to the command channel:

```
print#15, "u:"
```

In a future issue we'll be publishing instructions for a switch that will add pre-power-up device number selection (ie. 8 or 9).

Drive 1, Are You There?

Single disk drives have one very distinguishing characteristic; there's no drive 1! If you want to test this from within a program, here's a simple procedure. Open the command channel and send an I1 (initialize drive 1). Upon reading the error channel, the error returned will either be 0 for OK, or 21 for read error. This suggests there IS a drive 1, but there may or may not be a disk mounted. If a single drive is connected, the error will be 74, drive not ready.

This tip courtesy of Greg Beaumont at RTC. You might want to include this with the Catstrapolator utility later this issue.

SuperPET Bits

How many bits in a SuperPET? Seems like a silly question, doesn't it. Well, it is. The answer will be sure to impress your friends though. . . we'll have it here next issue, but you may want to add it up just for fun.

Just one preliminary note. . . Waterloo now has the 6502 Assembler available for the SPET. No word on price, call Waterloo.

The rest of our Bits and Pieces comes from the Computer Systems Group at Waterloo. Many thanks for permission to reprint.

Index Expressions In APL

An error in the evaluate of the individual components of an index expression cac cause errors or incorrect results when using WLU microAPL with a fragmented workspace. The problem exists only with Version 1.1. The expression evaluates incorrectly when a garbage collection occurs during the evaluation. The likelihood of this happening can be reduced by forcing a garbage collect (eg. M ⊏ □ WA) from time to time. The problem will never occur if the individual expressions are assigned to variables prior to performing the index operation. For example:

```
x ← a [ 1+1 ; 2+2 ]
```

changes to:

```
a1 ← 1+1
a2 ← 2+2
x ← a [ a1 ; a2 ]
```

Form Feeds and SuperPET Printer Output

Normally a SuperPET printer is dedicated to the SPET to which it is connected. It is up to the person using it to ensure that the printer is positioned to the start of a page before printing a listing. If the printer is connected to several SuperPETs through a device such as a MUPET, it can be shared by more than one person. It can even be physically distant from one or more of the micros to which it is connected, and adjusting the printer before producing a listing can be rather cumbersome. In response to requests from SPET users who wish to share printers, the following program was written:

```
010 open #2, "ffpatch,prg", output
020 loop 030    read i
040    if i=999 then quit
050    print #2, chr$(i);
060 endloop 070 close #2
080 data    5, 192,    0,  41,    0,    0
090 data  134,    8, 183,    5, 149,   15
100 data   50,   57,    5, 202, 174,   98
110 data  230,    2, 193, 130,   38,   22
120 data  204,    0,   96,   52,    6, 236
130 data  100, 189, 192, 114, 204,    0
140 data   12, 237, 228, 236, 100, 189
150 data  211, 123,   50,   98,   57,    0
160 data    0,    0,    0,    2,    0, 999
```

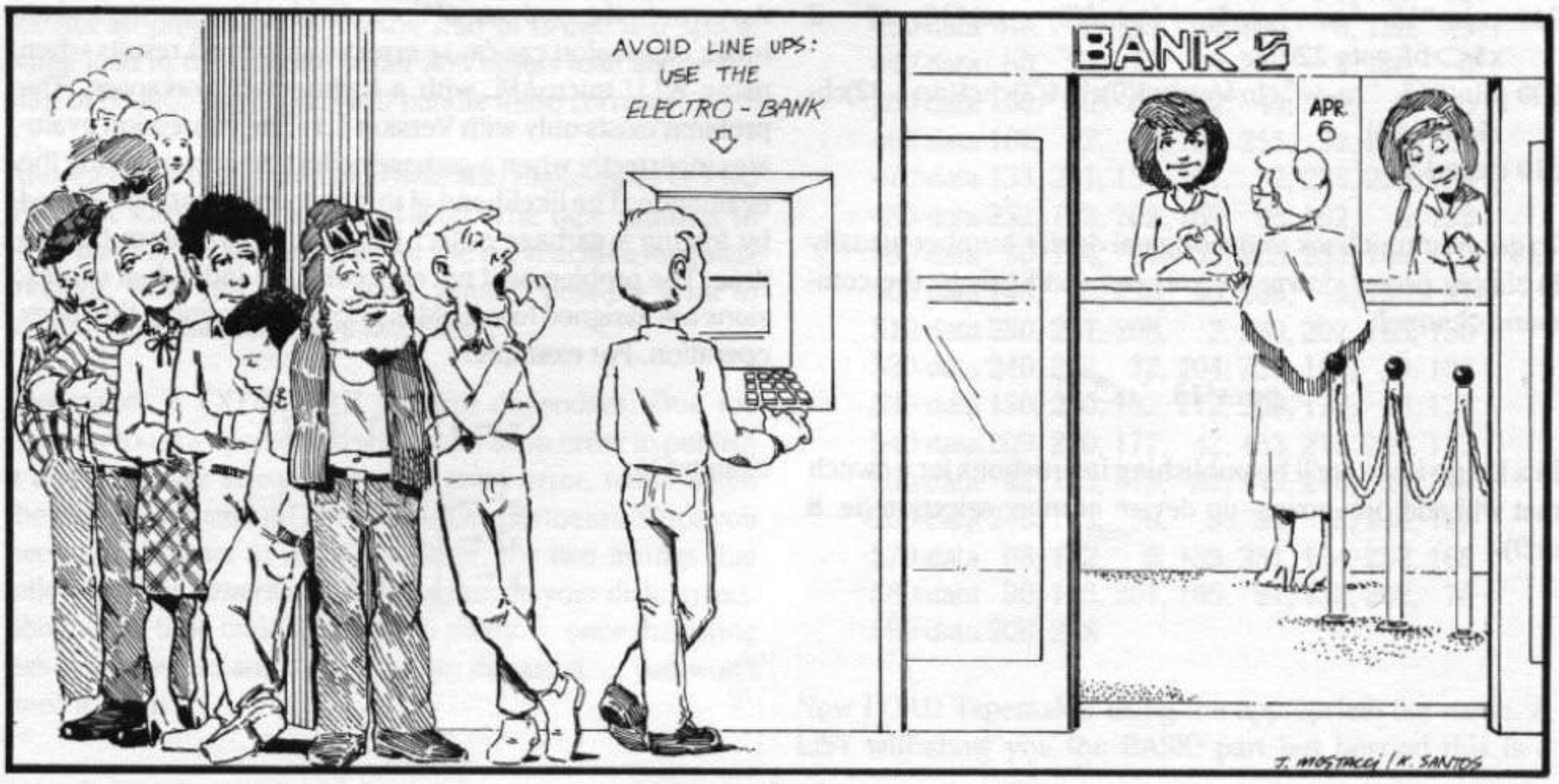
The program is run only once. It creates a disk file called "ffpatch" (form feed patch). The patch program is run from the SPETs main menu by entering "disk.ffpatch" when the SPET is turned on, or whenever a switch from 6502 to 6809 mode is made. When it has completed the program returns to the menu. The program applies a patch to the system routine called I3ECLOSE_ so that a FormFeed character is output whenever a printer file is closed. This will prevent printer output from more than one source from appearing on any single sheet.

Simulating a GET in PASCAL

The program below simulates a BASIC GET statement to read a character from the keyboard:

```
program main ( input , output );
 var
  c : char;
  io : file of char;
 begin
  writeln ( 'Character?' );
  reset ( io , 'keyboard' );
  c := chr(0);
  while (c = chr(0)) do
      read ( io , c );
  writeln ( 'Character = "',c,'"' );
 end.
```

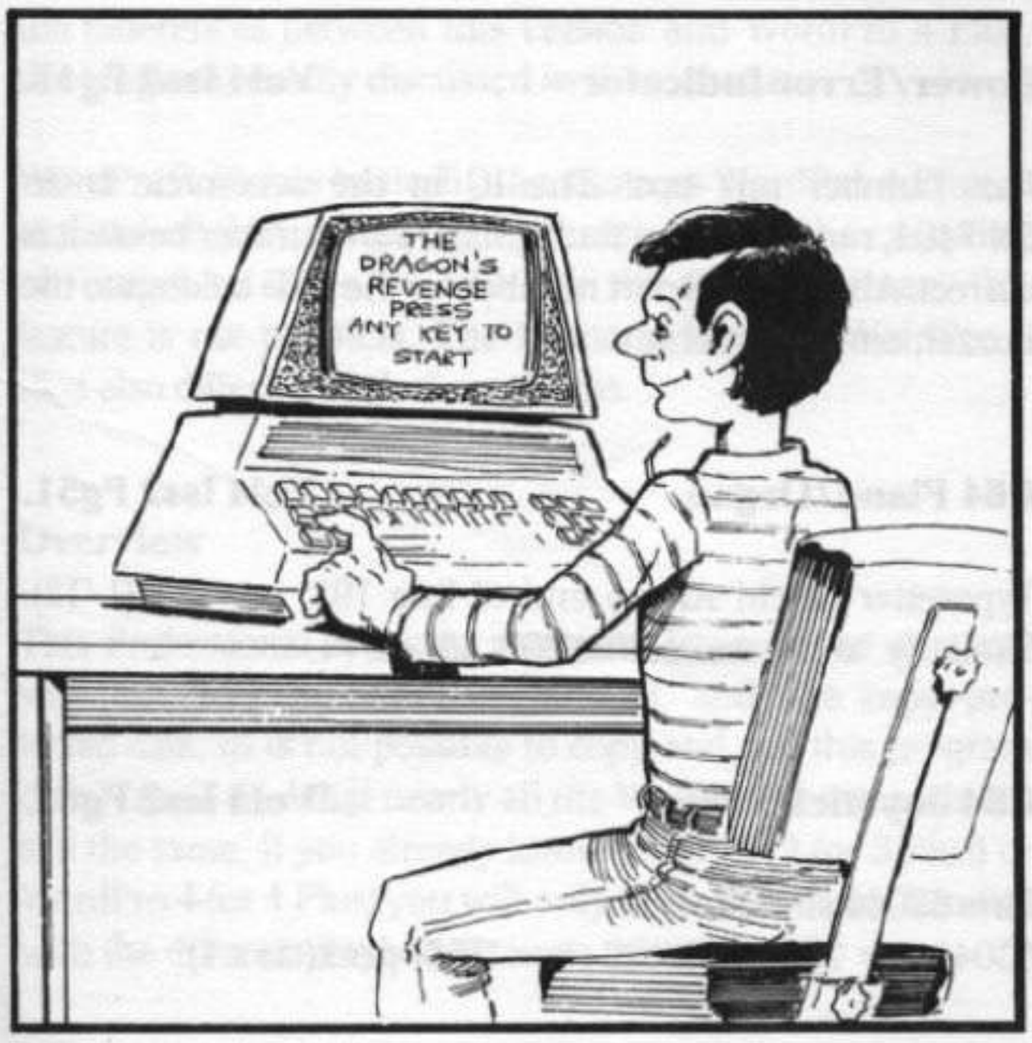
Until a key is hit, the read statement returns a null value in the variable C. When a non-null value is returned, the while-loop is terminated and the character is displayed by the second writeln statement. The character entered is not echoed directly on the screen when it is typed.



CompuKinks.

You shouldn't have rear-ended that guy like that.

It was a program I found in The Transactor.



Transbloopors

This section, although hopefully not regular, will cover publication errors and/or improvements for articles in previous Transactors. If you find a significant mistake in our mag, please let us know so we can pass it on. Ed.

C64 To PET/CBM **Vol4 Iss2 Pg12.**

Missing link! The 6 step procedure for loading a C64 program into a PET needs one extra step. It seems that when line 0 is deleted (step 5) the PET doesn't think that the size of ANY line could possibly be over 255 bytes. Therefore, the high order byte of the End–of–BASIC pointer is not adjusted properly. A SAVE at this point would store an extra 1K of waste. To fix it, add step 5.5:

```
5.5 poke 43, peek(43)–4 : clr
```

This will adjust the End–of–BASIC pointer and the CLR cleans up any other pointers.

Stack (Crackle) Pop **Vol4 Iss2 Pg13.**

Fooled! The C64 and VIC 20 SYS commands to simulate a POP, don't work. At first, the C64 SYS appearing to be working, but on further testing it was fubar. I didn't have a VIC 20 at the time and just assumed it would be the same plus 8192, but of course this didn't work either. The BASIC 1.0, 2.0, and 4.0 SYS calls are correct.

Using the SYS's from the earlier BASICs, I compared disassemblies and found the equivalent C64 and VIC 20 ROM code lies at:

```
64: sys 42622 ($a67e)
20: sys 50814 ($c67e)
Note: difference is 8192 ($2000)
```

But for some reason these don't work either. Argh! The results appear the same as RUN/STOP – RESTORE. I examined the code that performs SYS and although it's different than in earlier machines, it all looks pretty straightforward. But no go. Would someone like to take a stab at this and let me know what's happening? The code for SYS starts at E12A in the 64, and E127 in the 20.

Bugz **Vol4 Iss2 Pg19.**

Retraction! 1540 and 1541 diskettes ARE write interchangeable. The compatibility problem lies in the drive itself. "High profile" drives are not compatible with "low profile" drives. When the new 4040s and 2031s come out with low profile drives, they'll all be write compatible.

Bugz **Vol4 Iss2 Pg21.**

Misquoted! It was actually Don Lekei of North Vancouver BC that found the fix to the renouned C64 screen editor crash. Thanx Don.

Determining Screen Size **Vol4 Iss2 Pg37.**

Typesetter glitch! Those upside–down exclamation marks should be '=' signs and the double plus signs should be single plus signs. Most of those familiar with source code listings will have caught this error.

Power/Error Indicator **Vol4 Iss2 Pg41.**

Part number mix–ups! The IC in the schematic is an SN 7404, not 7400. The Radio Slack part number below it is correct. Also, The RS part number for the LED belongs to the buzzer, and vice–versa.

C64 Piano/Organ **Vol4 Iss2 Pg51.**

Typesetter glitch! At the end of line 193, co=2 ↑ (1/12). Similarly, at the end of line 200, a(i)=2 ↑ i.

C64 Joystick Ports **Vol4 Iss2 Pg61.**

Line 63040 should read:
63040 a = 255–peek(cia) : b = **255**–peek(cia + 1)

The WordPro Book Of Tricks

Donna Green
Commodore Canada

WordPro and The Commodore 64

Commodore Business Machines has recently announced the availability of WordPro for the Commodore 64 computer. This is the same famous program that runs on the 4032 (40 column) computer and therefore has almost the same name: "WordPro 3 Plus for the 64". For those of you who may recently have purchased this program, which is available from all Commodore dealers, we will explain here the differences between this version and WordPro 4 Plus, the program usually discussed in this column.

WordPro 3 Plus is basically the same as WordPro 4 Plus. It includes all the various editing functions with the exception of "output to video" – with the 40 column screen, this feature is not practical. The Numeric Mode in WordPro 3 Plus also differs slightly from 4 Plus.

Overview

This Professional Software program, is packaged as usual with an excellent reference manual, and one copy-protected disk. (It is not possible to copy and run this program disk.) You'll find that nearly all the WordPro commands are still the same. If you already know WordPro 3 (or 3 Plus) or WordPro 4 (or 4 Plus) you will only have to become familiar with the different keyboard to use this program.

There are a total of 329 lines available for main text with a minimum of 176. Together with the extra text area, which has a minimum of 23 lines, there is a grand total of 352 lines available on this 64 version. (Although the manual still recommends that this maximum number of lines be entered, I still recommend that 180 or 200 lines be entered to allow for more space in the extra text area. Another reason for selecting fewer lines, is that with WordPro 3 Plus and WordPro 4 Plus files being completely compatible – except for the length of text available, there is the potential problem of WordPro 4 Plus not being able to recall an entire 3 Plus file.)

Loading The Program

This program written for the Commodore 64 generally uses a 1541 disk drive and a 1525E matrix printer. Other disk drives and other printers, however, may be used if a "C64–Link" is attached. The program is loaded with the command load "*",8. (This obviously differs from the "Shift/Run" command for WordPro 4 Plus.) The series of questions that come to the screen should be answered individually, rather than pressing "Control" to default through all the selections. If a 1525E printer is attached, select "o" for Other, and "c" for CBM.

Other Printers And Disks

As many "64-owners" are aware, or as anyone will notice from advertisements in The Transactor, there is an attachment available called the "C64-Link" that will allow for other disk drives and printers to be connected to the computer. This means, that any compatible IEEE and parallel printers, as well as all disk drives can now be used with WordPro 3 Plus. For example, WordPro 3 Plus can successfully operate with an 8050 disk drive and an 8300P letter quality printer (Diablo)!

In order to run WordPro 3+ with the C64-Link, the "Link Relocator" program that comes with the C64-Link must first be loaded. This will bring to the screen a 5-item menu. It is then required to cursor down to the 5th item entitled "No Ram and Flipping" and to press return.

Three more selections must then be made. First the screen will display "Disk Configuration" with the selection to be made of either "Serial" or "IEEE". "Printer Configuration" next offers three choices: "Serial", "IEEE", or "Parallel". The final screen reads "Save to Disk with New Defaults: No/Yes". The computer is then ready to load WordPro 3+.

The Relocator program is now in the public domain and is included with all C64-Links.

New Features

Colour is Added!

One of the nicer features now added to WordPro, is the ability to change the colour of the text characters, the screen background, and the border.

Control + F1 increases the value of the colour for the characters.
Control + F3 increases the colour value for the screen, or background.
Control + F5 increases the colour value for the border.

Colour values may be decreased by pressing F2, F4 and F6 respectively.

Save Colours With Program

The colours of the screen, border, and characters may also be permanently saved with the program by entering a 13 line basic program. This program is listed on a covering page which is included with the program package.

The C64 Keyboard

1. Control Mode Has Three Control Keys!

"Control" mode can be entered with either the "Commodore LOGO" key, the "CTRL" key, or "F7" key. Any of these 3 keys will function as the "Control" key.

2. Backwards Slash Key Has a New Look!

The "English Pound" sign performs the same functions as the "Backslash" key in WordPro 4+. As you know, from other WordPro's, the Backslash can be used to turn on and off "shifted mode", it will turn the "beeper" on and off if the control key is pressed first, and it can also be useful for shortcuts in recalling and memorizing – by eliminating the need to type in the file name when the cursor is pointing to the name.

3. Tab Key Also Changes

The "back arrow" (← located just above the CTRL key) has the same function as the TAB key on other WordPros. Therefore, the back arrow would be pressed to tab over (move the cursor) to an indented area such as the beginning of a paragraph. Control + back arrow will jump the cursor either to the bottom of the text on the screen, or to a "block code", whichever comes first.

Differences In Numeric Mode

1. Setting Numeric Tabs

Numeric tabs are set the same way as regular tabs – Control + S. To enter numbers so that they will all right align, press Control + N before tabbing to the location of the first number. The "N" will highlight on the status line, confirming that "Numeric Mode" has been turned on. After all numbers have been entered, Numeric Mode is turned off by again pressing Control + N.

2. Totalling Columns of Figures

To total the columns of figures, it is necessary to tab over (using back arrow) to the first column of numbers, and to have the cursor on the line where the total is to appear. The cursor must be one column, or character position, to the right of the number, as illustrated below. The command "Control + equals sign" must be given, then the answer will appear!

Example:
Status Line Tab Marker: -

125←
25←
50←
(300)←
---←
(100)←
Cursor should be here □
when giving command to total

Note: Negative numbers may be indicated by parenthesis as shown, or with a minus sign after the number. Negative totals will appear in the answer, the way the negative number was entered in the text.

In this example, if only the bottom two numbers are to be totalled in the example above, cursor to the line that displays (50)←, and set a range (Control + r + return).

When the command to total is given, the status line will ask: "Range or All?". Press "r" for range and only the bottom two numbers will be totalled!

Similarly, if only the middle two numbers are to be added together, cursor to the line with 25, and set a range of two lines, Control + r + cursor down one + return.

When the command to total is given, the status line will again ask: "Range or All?". Again press "r" for range and only the middle two numbers will be totalled!

Special Comment On Erasing Function

WordPro 3 Plus, upon very close scrutiny, may appear to have one minor limitation – that being the inability to cancel an erase function. For example: If, after giving the command "Control + e(erase)". . . which is normally followed by "r" for remainder, or "a" for all. . . it was decided not to execute the erase function after all, it may seem that there is no way back to normal editing mode, without "erasing" some text on the screen. Pressing the Control key in this case, unfortunately, does nothing to cancel the command. While it is highly unlikely that this would occur often, it may be worthwhile to make note of this little trick – just in case.

There is just one option left in this situation, and this is the saving grace. By pressing the letter "l", after "Control + e", the system will return to edit mode! This is in fact, the latter half of the command to "Erase Lines in Range" (used to erase large quantities of text at once). The entire command actually involves setting a range of text to be deleted, of perhaps a paragraph or two, then erasing that range.

If no range has been set, and the command – Control + e(erase) + l(lines in range) is given, nothing really happens, except that a message appears on the status line "No Line Range Set". The appearance of an error message however, is much easier to live with than losing the "remainder" of text on the screen, or even worse "All" the text!

That's all for now! More next time.

The MANAGER Column

John Stoveken
Milton, Ont.

C64 Manager Arrives

Yes, The Manager is alive and well and running on the Commodore 64. The software has increased in power and flexibility and still maintains the straightforward friendliness of the current Manager.

The create options behaves very similarly with the power of color added to the screen image. You are now permitted up to 250 fields with lengths ranging from 1 to 40 characters. There is a limit to the total length of all fields which is dependant on the number of fields and size of display positions and ranges between 1000 to 3000 characters.

Speaking of display positions and arithmetic, we have completely revamped the math system and have implemented JPL, a programming language designed specifically for this database. JPL is a fully structured language that includes constructs such as:

```
if. . . . .then. . .else. . .
endif
```

and

```
while. . . . .do. . .
endwhile
```

JPL handles both numeric and character data to permit simple and boolean compare operations. As an example:

```
if (f1 = 'bert') and n3<100 then 'hi bert' to d1
```

will place 'hi bert' in display position one only if field one contains bert and field three has a value under 100. The data that JPL handles can be in fields, registers and display positions and all can be subscripted by others.

The ENTER/EDIT program now includes a full boolean search system as well as a real time 'calculate' mode. Although the standard search and hunt is available, the power of the boolean compares as in JPL are extended through to the report package as well.

The code is still hybrid BASIC and machine code (10 kbytes) which will permit the user full access to the code and data if he desires. All data fields are in the RE$() array, registers in the R() array and display positions in the DP$() array. The data file is a standard Commodore sequential file that can be accessed by BASIC or by The Manager's 'BASC' extended BASIC.

The .SCR File

The secrets of the screen shall be revealed this month.

Any data file that has been created with the Manager system has at least two disk files associated with it. One is the actual data file itself, called

```
'name          rel'
```

and the other is the screen file which is called

```
'name.scr      prg'
```

The actual data file contains ONLY information from the data record. Only the carriage return on the end of each record and the character 255 placed at the end of the data file make this file unique for The Manager. It is therefore quite easy to import a foreign data set into The Manager by producing a screen with proper fielding for the other data file.

The screen is a program file (created by create/revise) that loads into the computer's upper memory area ($6500/25856 to $7dff/32255) and this contains all the additional fielding, display and arithmetic information for the data file. After the screen is moved up from ram to the screen ram the system places information about the currently displayed screen into memory locations $7e00 to $7fff (32256 to 32767). The memory map is displayed in this table. The memory locations are given in hexadecimal notation.

Memory Location	Information
6500 – 68ff	–arithmetic storage area
6900 – 69ff	–screen number, line and column of display positions
6a00 – 6aff	–line,column and length of each field on screen one
6b00 – 6b80	–start position of each field on screen one in actual data record
6bfa	–graphics or upper/lower case screen
6bfb	–field # of index field if any
6bfc	–screen # of index field
6bfd	–length of index key
6bfe	–#.ind entries/#.mas entries
6c00 – 73ff	–screen one, including all text, arrows and backslashes as included during create –all display positions are placed here and this entire memory area is then transferred to the actual video ram
7400 – 74ff	–line,column and length of each field on screen two
7500 – 75ff	–start position of each field in the actual data record on screen two
7600 – 7dff	–screen two, including all text, arrows and backslashes as included during create

The following data is placed here when the screen is written to the video display.

7e00 – 7eff	–line,column and length of each field on the displayed screen
7f00 – 7f80	–start position of each field on the displayed screen in the data record

The following information is duplicated for each screen at 6bf0 and 75f0 respectively.

7ff0	–starting field number
7ff1	–# of fields on the screen
7ff2	–total record length
7ff3 – 7ff4	–high byte and low byte for the total number of records in the data file
7ff5	–current screen number
7ff6	–total number of screens
7ff8	–ending field number
7ff9	–# of fields on the screen
7ffa	–graphics or upper/lower case

ON REPORT GENERATE we have a small goodie that may help you along. When you are reporting registers or performing a calculation on numeric data (accumulating) the report system automatically allocates twelve characters to the numeric data. The data is right justified in the twelve character area with a leading space or minus sign. This works fine up to the time that you want to print ten numeric areas on a single eighty character line (120 is slightly larger than 80).

If you enter the fields in the standard left to right format, the following pattern will soon develop. . .

```
data one    : : : : : 1234.56
data two                  : : : : : -123.50
data three                              : : : : : : 952.12

print line  : : : : : 1234 : : : : : -123 . : : : : : 952.12 : : : : : : : : : : :
            1---5----1----5----2----5----3----5----4---
                      0         0         0         0
```

Since most reports would like to include all the data possible, the above print line does not prove adequate. If, instead of entering the data left to right, we specify the print positions right to left, we will not loose data underneath the overlaping spaces as seen below.

```
data one                              : : : : : : 952.12
data two                    : : : : : -123.50
data three  : : : : : 1234.56

print line  : : : : : 1234.56 : : -123.50 : : : : 952.12 : : : : : : : : : : :
            1---5----1----5----2----5----3----5----4---
                      0         0         0         0
```

TPUG Library Naming Conventions

Mike Donegan
Burlington, Ont.

Anyone who has been involved with microcomputers in the last year or two, has experienced the confusing problem of getting a disk of programs from a friend or club and trying to decide what he got. Frequently the name of the program is an obscure reference to the original application that only the author knows. By the time you receive the copy, it has been modified many times and is very cryptic.

I have lived with this problem for a few years. While living in western Canada, I regularly traded, mootched, begged, and borrowed programs from all I met. My personal collection soon grew to many disks. The problem was that when I went to get a program, I spent many hours just sorting through unrelated versions. I couldn't remember the special name that it had, or it was mixed up with versions written for a configuration I didn't have!

As one of the new librarians for TPUG, I find the problem has escalated. The club has 68 category disks, more than 20 monthly disks, and Commodore has over 50 educational disks. The club decided it was too much work for one librarian. The library now has three sections; Craig Bonner for VIC 20 programs, Jim Law for C64, and I'm handling PET/CBM. At our first meeting, we decided to use a number of conventions to help sort the programs and resulting disks.

Program Naming

New programs added to the library would use as the last two letters in the name, a period followed by a character to designate a machine or data type.

Disk Name and ID

The disk name would be followed by a period (like filenames) and a character to indicate the machine type or a special grouping of programs. The disk ID code would remain the same as before – the first character showing the subject category, followed by a digit for its number in the subject sequence.

List–Me Program

Finally, each disk would have a List–Me program, preferably as the first program on the disk. The contents would include the program directory plus a description of each program in REM statements.

TPUG and many other groups have tried to get some reasonable documentation on programs but programmers are a contrary and varied group. The only club that I know

with complete library documentation is WPUG (Winnipeg) mainly due to the diligent work of Bob Jolly. The problem is that the club doesn't have the resources to print multiple copies of 75 or more page directories.

The TPUG librarians would appreciate it if you would add these features to your programs, with any other documentation built into the program. Notes and instruction lists frequently get separated when they don't bear the same names.

The following are the codes TPUG will be using:

File Name Convention

program name.V		VIC Programs
	.C	C64 Programs
	.4	40 Column PET/CBM (9" crt)
	.F	Fat Forty PETs
	.8	80 Column CBM
	.P	General PET/CBM
	.S	SuperPET (SP9000)
	.B	B Series (available soon)
	.W	Word Processing Files
	.D	Data or Seq Files
	.Z	Misc. or Undefined
	.L	List-Me File

The program Copy-All for the C64 would be:

COPY-ALL.C

Disk ID Convention

Disk ID char	Ax	Assembler/Machine Code
	Bx	Business
	Cx	Communications
	Ex	Education
	Gx	Games
	Lx	Languages
	Nx	Math/Science
	Sx	Music
	Tx	TPUG Monthly disk
	Ux	Utility
	Zx	Miscellaneous

. . .where x is the Disk ID Number, 1-9, A-Z, with zero reserved for future use as a series directory.

Disk Name Convention

The disk name suffix (machine type) will be the same as program name suffixes with these additions:

.C	Commodore 64
.V	VIC 20
.P	PET/CBM
.S	SuperPET
.O	Old TPUG disks prior to March 83

The fifth assembler disk for 40 Column PET/CBMs would be:

Assembler #5.4 (with id:) A5

The List-Me file for this disk would be:

List-Me 4A5.L

The reason for choosing this format is that it provides for a simple method of documenting a disk that is compatible with all machine types.

If anyone has any ideas to improve the suggestions made here, I will be happy to hear them. But the overiding idea is *KISS* – Keep It Simple Stupid. The method has to be easy to remember and implement.

Sweet Sixteen. . .

Jim Butterfield
Toronto, Ont.

It's an accepted fact that eight bit computers are more powerful than four-bitters, and sixteen bit units more powerful than eight, and so on.

The question that bothers me is this: do more bits always mean greater cost-effectiveness? That's not the same thing as power: we're trading capability against price. More: we must consider not just the price of the microprocessor chip, but the price of the whole system.

I sometimes fear that we'll fall into the hi-fi fallacy. High fidelity was replaced by stereophonic systems. About ten years ago, a number of manufacturers said, "Well, if two speakers are better than one then four speakers must be even better yet". And thus a new quadrophonic industry – tapes, turntables, amplifiers, speakers and even quad headphones – was born. Or rather stillborn, since it never really caught fire. Quadrophonic systems really did sound better; but the improvement wasn't good enough in view of the substantial extra cost and space requirements.

What Is It?

We'll need to define terms. True sixteen-bit computers are those that access sixteen bits of memory in a single cycle. There are only a few microprocessor chips that do this: the best known are the 68000, the 8086 and the Z8000.

There's another class of machine that handles 16-bit information internally, but dips into memory eight bits at a time. These are often called pseudo-16-bit processors, and they are not new. The RCA COSMAC falls into this category, for example, and it's been around for many years. Some other pseudo-16's are the 6809 and the 8088 (Yes, the IBM Personal Computer is not a true 16-bitter!)

There's a major difference between true and pseudo. The ultimate speed of a machine depends on how many dips must be made to memory to perform a given task. This includes memory accesses for both data and instructions. With eight bits, you get only half as many bits at a time; you need to go to memory more often.

But we're not fully following the 16-bit phenomenon if we simply track the strict definition. There are two other things colouring the picture: better instruction sets, and greater memory access.

Instruction Sets and Memory Addressing.

It is nice to have a chip with powerful extra instructions. The new chips will perform (limited) multiplication and division; they have more sophisticated compare and branch instructions, and a wide choice of addressing modes.

Memory addressing may be one of the most significant new features of the new chips. As RAM (random access memory) becomes less and less expensive, we start to notice that 32K or even 64K isn't enough. The new chips allow addresses that go far beyond this. It's odd, in a way, since 16 bits and 64K go naturally together – but we need more memory addressing.

In Our Context. . .

So switching to sixteen bits (or even pseudo) gives us two major advantages: speed and more memory availability. If these were free, we'd accept them gratefully. But there's usually a cost, and we'll need to decide if it's worthwhile for us.

How much do you need extra speed? You'll have to answer the question yourself, of course, but one question you might ask yourself is this: how much time do I spend waiting for the computer to do something? Don't include time waiting for printer and disk – their speeds are not related to the problem.

If you have a machine language program, chances are that the program activities will seem to be instantaneous. You may need to slow down the whole process in order to read the screen. If you think about this, you'll see that in this case a super-speed chip would give you nothing.

That's not always the case. Some types of calculation take up lots of processor time – sorting is the most obvious application – and you'll find that extra speed pays off. You must decide whether your usage would benefit from the speedup.

Memory availability – not a true 16 bit characteristic, but one that often goes with the new chips – is also something that the user must evaluate. There are ways of getting extra memory on conventional systems – the SuperPET and the 96K PET show how it's done. But built-in extended addressing can be quite useful, if you need it. Do you need it?

Summary.

Sixteen bit systems cost more. It's not just the processor chip, but the extra memory and board wiring. You will often need extra memory for the same job, since flags will typically take up a sixteen-bit word instead of an eight-bit one (one bit is all that's really needed, but using a word is often more convenient).

It's up to you to decide if the extra price buys you anything.

I like to think that data falls into natural chunks. For calculators and similar operations, you need a four-bit machine; a decimal digit fits nicely into four bits. For text applications, eight bits is right for fitting an alphanumeric character. But I can't think of any natural data that sizes to sixteen bits. Not numeric variables – they need more than sixteen. Now when the thirty-two bit machines arrive, they will be a mathematician's delight. But sixteen bits don't quite make it. Yes, they'll give more power and speed – but I can't see them bringing in a new class of applications.

One last note. The eight-bit microprocessor is heavily underpinned by a major industry; the personal computer industry accounts for a very small part of micro purchases. The major financing comes from – no, not the military – but the video game industry. I wonder if the sixteen-bit chips will ever get comparable volumes – and thus comparable economics.

VIC 20 Screen Centering

Peter J. Lear
Burlington, Ont.

Are you tired of turning on your VIC and television and not seeing the whole picture? You could open up either the VIC or TV and make internal adjustments, but not too many people feel comfortable doing this. The alternative? Tell VIC to move over!

That's fine you say, but how? Well, the VIC has two memory locations just for this. The vertical centering is controlled by location 36865 ($9001) and the horizontal by 36864 ($9000).

What value do you put in these locations? Well, that's difficult to say as every VIC will match up differently with the various televisions on the market. You could arbitrarily PEEK and POKE to find the proper values, but this is cumbersome. Besides, you might end up losing all of the screen. Don't worry though, this effect is not permanent. Simply use the RUN/STOP RESTORE key sequence and the screen will return to its power up position.

Or you can save yourself all that trouble by entering Program 1. Once done and checked for accuracy (you may want to save it first), RUN it and use the cursor control keys (including shifted controls) to adjust your screen. When satisfied with the position of the screen, hit "HOME" and the program will give you back the values for your particular setup. Next time you turn on your VIC (or hit RUN/STOP RESTORE), just key in these values and your screen will be centered.

If you don't care for this, try Program 2. This is the same thing, only in machine language. It moves BASIC up and therefore is accessible whenever needed. All you need do is reference the chart below to initiate the machine language subroutine. It does not report the position values as the program is always available.

To enter Program 2A you can use any of the machine language monitors for the VIC, or use a PET/CBM with the built in monitor and enter Program 2B. In the interest of standardizing things on the VIC, remove any memory cartridges before typing it in. You can do it with extra memory, but things change with extra memory that will affect this procedure. Once it has been SAVEd properly, any VIC with or without extra memory can utilize the program.

If you're using a VIC, enter the monitor with SYS 13 and type:

m 002b 0032 <return>

Change the values shown to read:

002b 01 10 e8 10 e8
0030 10 e8 10 07 1b

This procedure sets the End of BASIC pointer such that a SAVE will store our program correctly. Now type in Program

2A. Exit the monitor and check for errors with the following line:

```
for i = 4096 to 4327 : x = x + peek(i) : next : print x
```

You should get a value for back of 21786. If not, re-enter the monitor and look for errors. Once correct, SAVE it as a normal program with:

```
save "screen center"
```

Verify it and you're ready to run.

If entering on a PET/CBM, go into the monitor with SYS 4 and type:

```
m 0028 002e        <return>
```

Change the given values to read:

```
0028 01 04 e8 04 e8 04 e8 04
```

Once again, these steps set up the proper pointers for SAVE later on. Type in Program 2B, exit the monitor, and use the following to check for errors:

```
for i = 1024 to 1255 : x = x + peek(i) : next : print x
```

If you do not get back 21762, re-enter the monitor and check for errors. When the program is all clean, SAVE it. You can now try it out on a VIC.

Table 1

VIC Setup	To Access:
no cartridges	SYS 4230
3K cartridge	SYS 1158
8K or more	SYS 4742

Program 1: BASIC Version

```
100 rem vic 20 screen centering
110 rem by peter lear
120 hz=5 : vt=25
130 poke 36864, hz : poke 36865, vt
140 if peek(197)=23 and peek(653)=0 then hz=hz+1 : if hz=64 then hz=0
150 if peek(197)=23 and peek(653)=1 then hz=hz-1  : if hz=-1 then hz=64
160 if peek(197)=31 and peek(653)=0 then vt=vt+1  : if vt=128 then vt=0
170 if peek(197)=31 and peek(653)=1 then vt=vt-1  : if vt=-1 then vt=128
180 if peek(197)=62 then 200
190 goto 130
200 for i=1 to 10 : get a$ : next : rem clear keybd queue
210 print " to center screen: " : print
220 print " poke 36864, " ;peek(36864)
230 print " poke 36865, " ;peek(36865)
```

Program 2A: VIC 20 Hex Dump

```
.m 1000 1060      <return>
.: 1000 00 3c 10 64 00
.: 1005 99 22 93 11 11
.: 100a 11 11 11 11 11
.: 100f 11 11 1d 1d 1d
.: 1014 1d 1d 53 43 52
.: 1019 45 45 4e 20 43
.: 101e 45 4e 54 45 52
.: 1023 22 3a 99 22 11
.: 1028 1d 1d 1d 1d 1d
.: 102d 42 59 20 50 45
.: 1032 54 45 52 20 4c
.: 1037 45 41 52 22 00
.: 103c 56 10 6e 00 9e
.: 1041 28 c2 28 34 33
.: 1046 29 aa 32 35 36
.: 104b ac c2 28 34 34
.: 1050 29 aa 38 37 29
.: 1055 00 00 00 e6 2c
.: 105a e6 2e e6 30 e6
.: 105f 32 a9 01 65 2c

.m 1064 10c3      <return>
.: 1064 a9 4e 8d 77 02
.: 1069 a9 45 8d 78 02
.: 106e a9 57 8d 79 02
.: 1073 a9 0d 8d 7a 02
.: 1078 a9 04 85 c6 c6
.: 107d 2b a9 00 a2 00
.: 1082 81 2b e6 2b a2
.: 1087 ff a4 ff 88 d0
.: 108c fd ca d0 f8 a4
.: 1091 c5 ae 8d 02 e0
.: 1096 01 d0 26 c0 17
.: 109b d0 0e ad 00 90
.: 10a0 c9 00 d0 02 a9
.: 10a5 40 e9 01 8d 00
.: 10aa 90 c0 1f d0 34
.: 10af ad 01 90 c9 00
.: 10b4 d0 02 a9 80 e9
.: 10b9 01 8d 01 90 10
.: 10be c7 c0 17 d0 0e
.: 10c3 ad 00 90 c9 40

.m 10c8 10e7      <return>
.: 10c8 d0 02 a9 00 69
.: 10cd 01 8d 00 90 c0
.: 10d2 1f d0 0e ad 01
.: 10d7 90 c9 80 d0 02
.: 10dc a9 00 69 01 8d
.: 10e1 01 90 c0 3e d0
.: 10e6 9f 60 00 00 00
```

Program 2B: PET/CBM Hex Dump

```
.m 0400 0478      <return>
.: 0400 00 3c 04 64 00 99 22 93
.: 0408 11 11 11 11 11 11 11 11
.: 0410 11 1d 1d 1d 1d 1d 53 43
.: 0418 52 45 45 4e 20 43 45 4e
.: 0420 54 45 52 22 3a 99 22 11
.: 0428 1d 1d 1d 1d 1d 42 59 20
.: 0430 50 45 54 45 52 20 4c 45
.: 0438 41 52 22 00 56 04 6e 00
.: 0440 9e 28 c2 28 34 33 29 aa
.: 0448 32 35 36 ac c2 28 34 34
.: 0450 29 aa 38 37 29 00 00 00
.: 0458 e6 2c e6 2e e6 30 e6 32
.: 0460 a9 01 65 2c a9 4e 8d 77
.: 0468 02 a9 45 8d 78 02 a9 57
.: 0470 8d 79 02 a9 0d 8d 7a 02
.: 0478 a9 04 85 c6 c6 2b a9 00

.m 0480 04e0      <return>
.: 0480 a2 00 81 2b e6 2b a2 ff
.: 0488 a4 ff 88 d0 fd ca d0 f8
.: 0490 a4 c5 ae 8d 02 e0 01 d0
.: 0498 26 c0 17 d0 0e ad 00 90
.: 04a0 c9 00 d0 02 a9 40 e9 01
.: 04a8 8d 00 90 c0 1f d0 34 ad
.: 04b0 01 90 c9 00 d0 02 a9 80
.: 04b8 e9 01 8d 01 90 10 c7 c0
.: 04c0 17 d0 0e ad 00 90 c9 40
.: 04c8 d0 02 a9 00 69 01 8d 00
.: 04d0 90 c0 1f d0 0e ad 01 90
.: 04d8 c9 80 d0 02 a9 00 69 01
.: 04e0 8d 01 90 c0 3e d0 9f 60
```

Catstrapolator

Writing disk based software? Often you'll want to extrapolate some information from your directory or Catalog – hence the name "Catstrapolator".

This might be disk name, ID, file type, or even blocks free. Programs to obtain this information from the disk unit have been published in the past, but they usually read it directly from disk memory. With so many disk operating systems released, a new version was necessary for each. This one is universal.

Catsrapolator goes after the disk directory through Secondary Address 0. SA 0 is reserved by the DOS for program loading and is also used for "loading" the directory (ie. LOAD "$",8). When the DOS sees a $ sign as the first character of the filename in a LOAD command, the information in the directory is manipulated by the DOS into a format compatible with LIST. Thus, the command:

```
OPEN 1, 8, 0, "$0"
```

causes the disk to deliver catalog data the same way it would for LOAD "$0",8.

Fortunately, Commodore has maintained a fairly consistent catalog format which makes Catstrapolator universal for all drives including 4040, 2031, 8050, 8250, 9060/90, and 1540/41. The only non–universal part is the string input routine held in the data statements at the start of the program. This is actually Jim Butterfield's Universal String Thing but even this is not universal between VIC/64s and PET/CBMs. If you're already using this utility somewhere else in your program, only lines 190 onward will get slipped in. Otherwise, VIC/C64 users will replace lines 130–170 with the separate sub–listing shown.

Program Operation

Catrapolator will work all by itself as shown, but it was meant to be used as a subroutine. You can even remove parts that return information you may not be interested in, such as file type (ie PRG, SEQ, etc.).

Line 180 should be included with the subroutine so that it is executed on each call. Make sure none of these variables are used elsewhere in your program so as not to disturb them.

Lines 190–210 input information that would normally be passed to the routine, in which case 190–210 would be omitted. FI$ is the target filename. This is transferred to PT$ for the OPEN command. However, if FI$ is null, PT$ is set to "no pattern", a filename that probably does not exist, and only blocks free will be reported. If FI$ IS passed, BU will be set to the number of blocks used by FI$, but only if it is in the directory. If no match is found, BU will remain zero.

FI$ may even contain pattern matching characters. For example, if FI$ = "CAT*", all disk files starting with CAT will be reported, BU will return the blocks used by the last match, and B1 will be the number of blocks used by the first match. Likewise, F$ and FT$ will be the filename and the file type last matched with FF$ and F1$ containing the filename and type of the first match. BF will always return Blocks Free.

Modifications

Lines 240–300 return the disk name and ID. If this is the only info you'll need to extrapolate, simply omit lines 310–540.

Several other features of this subroutine can also be skipped if they won't be necessary or just to reduce space consumption. If you won't be concerned with blocks used by a file, it's actual filename (if pattern matching characters are present) and filetypes, then omit 360–460, 490, & 500 and move line 540 up to 520. This variation will only get back Blocks Free. Many combinations are possible. . . you can decide which you need most.

You may have noticed two tests on ST, the status variable. If the program is entered in its entirety, the test at line 480 can be left out. ST will be caught in 390 unless 390 is removed.

Technicalities

Those POKEs scattered about the program are changing the machine code itself. You'll notice that the POKEs are directed right at the machine code, where the test for "delimiter received?" is performed. By changing the delimiter character, the program can inch its way along from one part of the directory to the next.

The delimiter is initially a Carriage Return as set up in the machine code. Line 240 changes it to a quote. When the input routine is called in 250, the characters up to and including the first quote in the disk header are essentially "stripped off". Immediately following is the Disk Name which is retrieved by the second SYS MC.

After the ID is obtained, the delimiter is changed to a zero since this will mark the end of the header line.

The Blocks Used by specific files are read by two GET#s after the line links are stripped (also with two GET#s). Following this, the delimiter is changed back and forth from 34 (quote) to zero for reading filenames and filetypes.

Before exiting, the delimiter is set back to Carriage Return for later use.

Summary

Using this subroutine can be invaluable to Business software programmers. A truly smart program could determine if there's enough room on the disk to update a file by comparing the number of blocks it uses to Blocks Free. Also, a program could decide if the correct diskette has been inserted by examing the ID or disk name.

Admittedly, Catstrapolator is dependent on Jim's machine code input routine, but no doubt, String Thing could find itself another use in a different section of the host program.

Finally, most of the PRINT statements will probably be removed when the subroutine is inserted, making it much shorter than it looks. You might also combine several of the single lines.

String Thing – VIC 20 / Commodore 64 Version

```
130 data 160,   2, 177,  45, 153, 137,   0, 200, 192,   6, 208, 246, 162,   1
140 data  32, 198, 255,  32, 228, 255, 201,  13, 240,  15, 164, 142, 145, 140
150 data 200, 132, 142, 196, 139, 240,   4, 165, 144, 240, 234,  76, 204, 255
160 restore : for j = mc to mc+41 : read x : poke j, x : t = t+x : next
170 if t<>6120 then print " typo in data " : stop
```

```
100 i$ = "abcdefghijklmnopstuvwxyz1234" : rem i$ must be 1st var used
110 i$ = i$ + i$ + i$                   : rem establish input buffer
120 mc = 896                            : rem mach code start addr
130 data 160,   2, 177,  42, 153, 184,   0, 200, 192,   6, 208, 246, 162
140 data   1,  32, 198, 255,  32, 228, 255, 201,  13, 240,  11, 164, 189
150 data 145, 187, 200, 132, 189, 196, 186, 208, 238,  76, 204, 255
160 restore : for j = mc to mc + 37 : read x : poke j, x : t = t + x : next
170 if t <> 5767 then print "typo in data" : stop
180 z$ = chr$(0) : pt$ = "no pattern" : b1 = 0 : bu = 0 : ff$ = "" : f1$ = ""
190 input "display blocks used by file: " ;fi$
200 input "                  on drive: " ;dr
210 input "                on device: " ;dv
220 if fi$ <> "" then pt$ = fi$
230 open 1, dv, 0, "$" + mid$(str$(dr),2) + ":" + pt$    : rem sa must = 0
240 poke mc + 21, 34                                     : rem make input delimiter a quote
250 sys mc : sys mc                                      : rem strip to 1st, input to 2nd
260 dn$ = left$(i$,peek(189))                            : rem disk name string
270 get#1, a$, a$, b$                                    : rem first a$ is waste
280 id$ = a$ + b$                                        : rem disk id = next two
290 print "disk name = ";dn$
300 print "  disk id = ";id$
310 poke mc + 21, 0                                      : rem make input delimiter 0 (end of line)
320 sys mc                                               : rem strip rest of head
330 get#1, w$, w$                                        : rem 2 waste bytes
340 get#1, a$, b$                                        : rem file blocks or blocks free
350 bf = asc(a$ + z$) + asc(b$ + z$) * 256               : rem calculate block count
360 if b1 = 0 then b1 = bf                               : rem blk cnt for 1st match set once only
370 poke mc + 21, 34                                     : rem make input delimiter a quote
380 sys mc                                               : rem strip to first quote
390 if st then 520                                       : rem no quote befor blks free, must b end
400 sys mc                                               : rem input up to next quote (filename)
410 f$ = left$(i$,peek(189))                             : rem 189 holds length of valid input
420 if ff$ = "" then ff$ = f$                            : rem ff$ = first filename match
430 poke mc + 21, 0                                      : rem make input delimiter 0 again
440 sys mc                                               : rem get remaining to extract
450 ft$ = right$(left$(i$,peek(189) - len(str$(bf))),3)  : rem file type
460 if f1$ = "" then f1$ = ft$                           : rem f1$ = file typ for 1st match
470 get#1, w$, w$                                        : rem waste two more
480 if st then 520                                       : rem end of file, go print blks free
490 bu = bf                                              : rem . . .only if file match found
500 print "blocks used by "ft$" file "f$" = ";bu
510 goto 340
520 if bu then print "blocks used by 1st "f1$" file "ff$" = ";b1
530 rem print above only if bu was set (blocks used)
540 print "blocks free = ";bf
550 close 1                                              : rem close dir
560 poke mc + 21, 13                                     : rem make input delimiter cr again
570 end : return                                         : rem remove end for subrout use
```

Making Friends With Sid Part 2

Paul Higginbottom
Toronto, Ont.

Hello again. Last issue, we got acquainted with some of the various terms and parts of the SID chip. We're now equipped to learn some more advanced things about the synthesizer, as well as more advanced techniques to fully utilize what we have already learned, to, for example, produce more than one note simultaneously, and then, to create software that can play actual pieces of music.

So there are, in fact, two areas that this and subsequent articles plan to deal with:

1) Define the capabilities of SID

2) Explain some software techniques to make SID perform.

This time, I'd like to put some of the last article's theory into practice, by giving some parameters for the SID, which will make it sound similar to musical instruments. I think this would be useful, so that you will be able to see that a music synthesizer is not limited to beeps and pops, and other sounds from television shows like "The Twilight Zone"!

In the last article, the various parameters of a voice were outlined, except for the filter in the SID. Essentially, the filter will (as is implied) filter the sound output from any of the voices in a number of ways. The actual term 'filtering' means that the sound is changed by quietening the voice to varying degrees above, below or around a given 'CUTOFF' frequency. However, don't worry about understanding this concept fully yet, since this issue won't use the filtering capabilities of the SID. I simply wanted to make you aware of this feature in the SID, so you won't be taken by surprise in the future!

To begin with, let's try to emulate one of the simplest sounds: A piano. When a piano key is struck, the sound begins immediately, and then fades away in about two seconds if the key is held down. If the key is released before the sound has faded away, it will fade much more rapidly, in say, half a second.

Try this program:

```
 10 sid=54272
 20 for i=0 to 24 : poke sid+i, 0 : next
 30 poke sid+24,15
 40 poke sid+5, 10
 50 poke sid+6, 9
 60 key=197
 70 poke sid+1, 16
 80 get a$ : if a$="" goto 80
 90 poke sid+4, 33
100 if peek(key)<>64 goto 80
110 poke sid+4, 32
120 goto 80
```

Explanation of Program:

Line 10 declares the variable SID to the start location of the SID chip.

Line 20 POKE's all the SID locations with a zero to initialize the chip.

Line 30 sets SID register 24 to 15, which sets the chip to maximum volume.

Line 40 sets SID register 5 to 10, which makes the attack of voice one 0, and the decay value 10.

Line 50 sets SID register 6 to 9, which makes the sustain of voice one 0, and the release value 9.

Line 60 declares the variable KEY to the zero page memory location which holds the keyboard matrix number of the current key depressed, or 64 if no key.

Line 70 sets SID register 1 to 16, which sets the high order byte of the frequency of voice 1 (therefore, frequency of voice 1 = 16*256) (see last article for explanation of 'low' and 'high' bytes).

Line 80 waits for a key, by GETting a keypress from the keyboard, and IF the keypress is a null, ie., no key has been pressed, the program will GOTO the same line and keep waiting.

Line 90 sets SID register 4 with 33, which gates voice 1 on with a triangular waveform (see last article for explanation [33=32+1])

Line 100 checks to see if a key is still depressed (as with a piano), and if it is (ie., location KEY is still something other than 64), the program will GOTO the same line and check again.

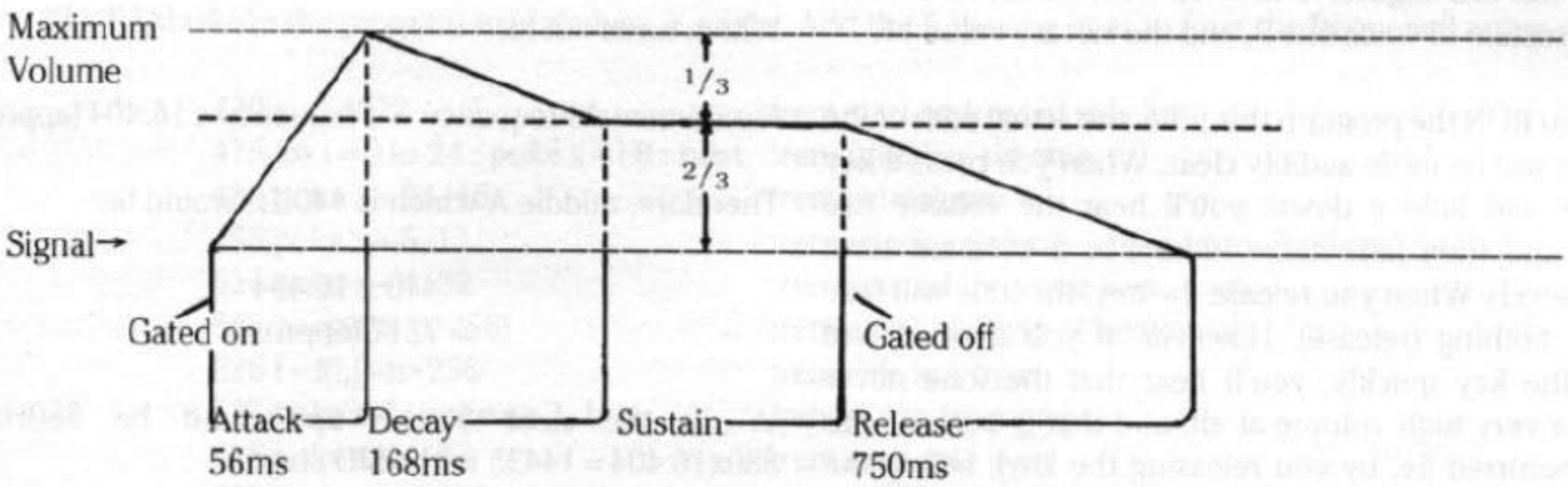
Line 110 sets SID register 4 with 32, which gates voice 1 off now that no key is depressed on the keyboard, still with a triangular waveform.

Line 120 simply goes back to line 80 to allow the program to continue indefinitely (to stop the program, the STOP key must be pressed).

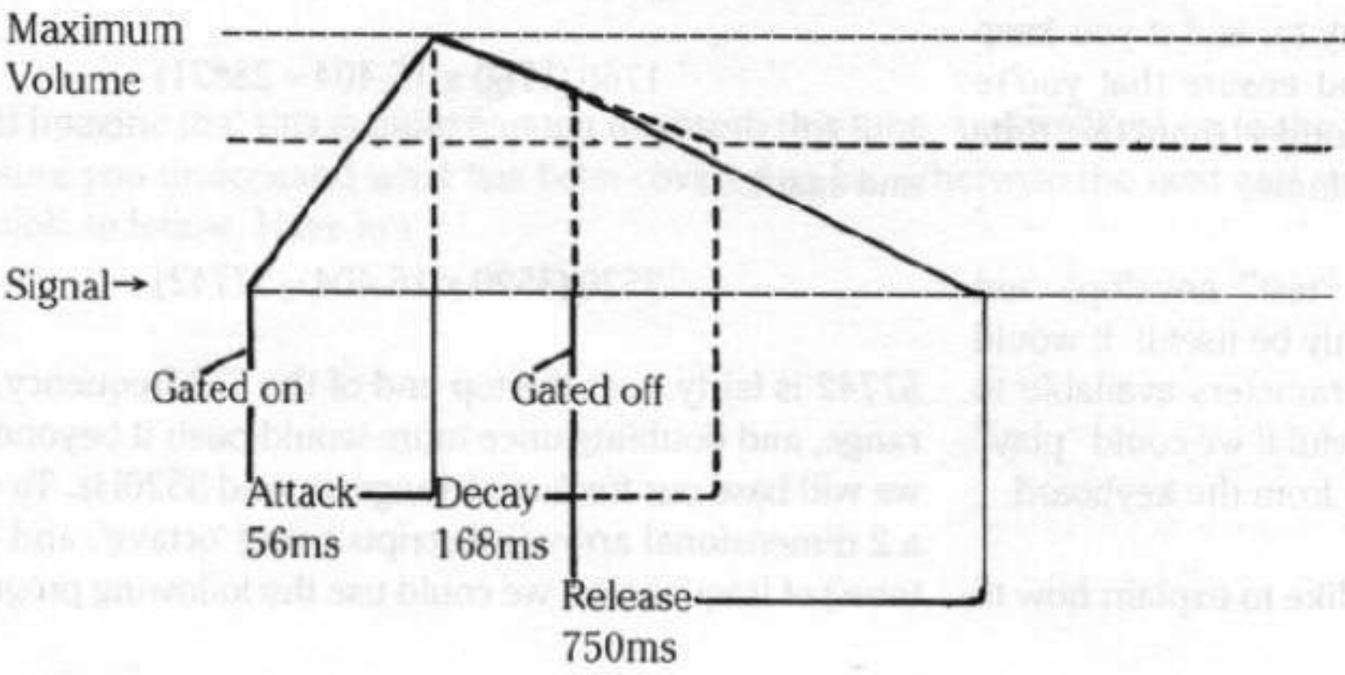
Something which ought to be understood here, is the fact that when a voice is gated off, ie. released, the envelope RELEASEs from WHEREVER it had reached. Probably a diagram would be the best way to show this:

Example:

Attack	=5,	ie. 56 milliseconds.
Decay	=5,	ie. 168 milliseconds.
Sustain	=10,	ie. two thirds of maximum volume
Release	=8,	ie. 750 milliseconds.



Now using the same parameters, only gating the voice off (ie. releasing) at a different point:



Here, it can be seen that the voice was gated off (ie. released), before the envelope had decayed to its sustained level, but when the voice was released, it simply released from the point it had reached.

This relates to the program you just entered, because it uses this fact to simulate the 'feel' of a piano keyboard. That is, as soon as you release the key on the keyboard, the envelope will begin its release cycle which is set at 9, one less than the decay value (10), giving the same response as a piano key, by fading away quicker once the key is released. Of course the force with which the key is hit in the first place, which on a piano gives the initial volume, cannot be simulated here, because a key on any computer keyboard, is either DOWN, or UP; the speed of transition cannot be detected.

Similarly, if a voice is gated on before the release cycle has finished, the attack will begin from wherever the envelope currently is (ie. the current output volume). To see this more clearly, enter the following to change our program:

```
40 poke sid+5,11*16+13
50 poke sid+6,9*16+11
```

Line 40 sets SID register 5 to 11*16+13, which makes the attack of voice one 11, and the decay value 13.
Line 50 sets SID register 6 to 9*16+11, which makes the sustain of voice one 9, and the release value 11.

When you RUN the program this time, the actual path of the envelope will be more audibly clear. When you press a key this time and hold it down, you'll hear the volume rise (attack), and then fade some (decay) to a constant level (sustain level). When you release the key, the tone will fade away to nothing (release). However, if you depress and release the key quickly, you'll hear that the tone never reaches a very high volume at all, and that is because the release occurred (ie. by you releasing the key), before the envelope had reached either its maximum, or the sustained level of volume. Similarly, if you press the key again very soon after releasing it, you'll note that the sound builds up again from the level it had faded away to, and if you keep depressing and releasing the key, and ensure that you're holding the key down for slightly longer than the time you're not, you will "pump up" the volume.

A program that would allow us to "test" envelope and waveform combinations would certainly be useful. It would enable us to experiment with the parameters available to create a desired sound. It would be useful if we could "play" the Commodore-64's synthesizer too, from the keyboard.

Before we can get that far though, I'd like to explain how to derive the frequencies for musical notes on the Commodore 64.

In a musical scale, the ratio of pitch between one octave and the next is 2:1. If we had the frequencies of the 12 semi-tones of the top octave, we could generate the values of all the lower notes by continually dividing all 12 by 2 to derive the pitch of the semitones in the next octave lower. It is not necessary to go into the math here, but if the ratio between octaves is 2:1, then the ratio between semi-tones is $2 \uparrow (1/12):1$.

Middle A on a piano, is 440Hz. To convert harmonic frequencies to the fundamental frequencies we need to put in the SID registers, we need to multiply the former by a constant which is derived from the frequency of the internal clock in the SID chip, and the system clock.

$$\text{Frequency} = \frac{\text{Fundamental Frequency x system clock speed}}{\text{SID clock speed}}$$

Therefore:

$$\text{Fundamental Frequency} = \frac{\text{Frequency x SID clock speed}}{\text{system clock speed}}$$

Which turns out to:

Fundamental Frequency = Frequency x 16.404 (approx.)

Therefore, middle A which is 440Hz, would be:

= 440 x 16.404
= 7217 (approx.)

'A' in the next octave up would be 880Hz, or 880x16.404=14435 in the SID chip.

The maximum value in the SID chip is 65535 (255 in both the low and high byte), therefore, by doubling again. . .

1760 (1760 x 16.404=28871)

and again. . .

3520 (3520 x 16.404=57742)

57742 is fairly near the top end of the SID frequency value range, and doubling once more would push it beyond it, so we will base our frequency range around 3520Hz. To create a 2 dimensional array (subscripts being 'octave', and 'semi-tone') of frequencies, we could use the following program:

```
100 fr=3520                          :rem note 'a' in top octave
110 co=2↑(1/12)                      :rem constant multiplier for next semitone
120 for i=1 to 9 : fr=fr/co           :rem start fr at 'c' by going back 9 semitones
130 next
140 ss=16777216                      :rem sid clock
150 cs=1022730                       :rem cpu clock
160 fc=ss/cs                         :rem frequency multiplying constant
200 dim f(7,11)                      :rem frequency array (octave, semitone)
300 for i=0 to 11                    :rem cycle through 12 semitones
310 s=fr*fc                          :rem calculate sid value of semitone in top octave
400 for j=7 to 0 step-1 : f(j,i)=s : s=s/2
410 next                             :rem calc value for all 8 octaves
420 fr=fr*co                         :rem go onto next semitone
430 next                             :rem continue through all 12 semitones
450 rem
460 rem print out all the frequencies
500 print "frequency table"
510 print "---------------"
520 print "oct sem frequency"
600 for i=0 to 7
610 for j=0 to 11
620 print i;tab(4);j,int(f(i,j))
630 next j, i
```

The REMarks in the program explain how it works. Add the following lines to hear the frequency array:

```
470 s=54272                          :rem start address of sid chip
475 for i=0 to 24 : poke s+i,0 : next :rem initialize sid chip
480 poke s+24, 15                    :rem set volume
485 poke s+5, 11                     :rem attack=0:decay=0:sustain=0:release=11
624 poke s+4, 32                     :rem gate off the voice first
625 h=int(f(i,j)/256)                :rem calc high byte of frequency
626 l=f(i,j)-h*256                   :rem calc low byte
627 poke s, l : poke s+1, h          :rem put in frequency
628 poke s+4, 33                     :rem now gate it on
629 for k=1 to 100 : next            :rem wait a bit
```

When you RUN the program this time, as the frequencies are listed, each pitch will be sounded.

I would imagine that this is quite enough to absorb this time, and we'll get on to the parameter testing program next time. Make sure you understand what has been covered so far, otherwise the next and subsequent articles will slowly become impossible to follow. Have fun.

New Character Sets On The Commodore 64

Jim Butterfield
Toronto, Ont.

Question: How do I redefine the characters on the Commodore 64?

Answer: There are so many ways that it's not possible to give a general answer. The following outlines the steps to follow, with a specific example:

A. Plan where you want everything to go.

Screen memory, characters, and sprites if you use them must all fit into the same 16K block of memory. To do your own character set, you'll probably want to leave the default block, 0 to 16383. A good block to pick would be the top one, which extends from 49152 to 65535. We'll pick the detailed locations for characters and screen a little later.

Don't do this yet, but we will select the top block by performing POKE 56576, 4.

Now, within that block we must pick screen and character set locations. Looking at the memory maps, we find that the top half, addresses 57344 to 65535, is used for the Kernal ROM. We could still use this area: POKEing to the screen and the character set would still make the changes in RAM to cause the screen to change appropriately. The only problem might be that we couldn't PEEK these locations – we'd get the ROM instead of the RAM. Some programs like to PEEK the screen to see what's there.

Locations 53248 to 57343 are not too suitable – the I/O chips are located there, and we'd have to do extra work to PEEK or POKE. That leaves 49152 to 53247 for the screen and character set.

A character set (the "character base") must start at an even multiple of 1024, and of course must be in the selected memory block. This leaves us with 49152 (start–of–block plus 1024 times multiple 0) and 51200 (offset multiple 2). It doesn't matter which one we pick. . . let's use the first one, which means that our character set will go from 49152 to 51199. Now for the screen: we must pick any multiple of 1024 that's in range. Skipping over the character set area, we have a choice of 51200 (multiple 2) and 52224 (multiple 3). We'll pick the first, so that screen memory occupies 51200 to 52223.

Let's look at those multiple numbers again. A character set starting at 49152 is at an offset of zero within the 16K block. . . which is 0 times 1024. A screen memory block starting at 51200 is at an offset of 2048 within the block. . . which is 2 times 1024. To set the video chip to use these actual screen addresses we'll do some arithmetic on the multiples: 2 (the screen) times sixteen, plus 0 (the character set), giving 32. . . and we POKE 53272,32.

B. Tell Basic

The two POKEs have told the video chip where screen memory and the character set is located. . . but we must also tell Basic about it, or Basic will insist on putting everything in the old address, 1024 to 2032. We tell Basic about the screen with POKE 648, 200. Why 200? That's the screen memory address (51200) divided by 256.

It wouldn't hurt to clear the screen here, since we've moved to a new area of memory that's cluttered with unused values.

C. Supply a Character Set

We have told the video chip that there is a character set at 49152. . . but we haven't put any characters there. You can define your own characters, of course: for the moment, let's just copy the characters out of the ROM.

The processor will find the character ROM at address 53248 to 55295. That's odd for two reasons. First, the video chip found it at an entirely different location. . . we can dismiss this a one of the magics of electronics. Secondly, we normally see the I/O chips at these locations. . . the character set is hidden behind these chips, and we'll need to expose it by POKE 1, 51. . . but not yet!

Before we make the I/O chips vanish, we'll need to lock out the interrupt. The interrupt sequences, which check the keyboard and do other jobs, will get hopelessly confused if the I/O chips are not there. Take my advice: never get the interrupt sequences hopelessly confused; it's not good for your program's health. We may lock out the interrupt with the command POKE 56333, 127. . . but we must remember faithfully to put it back.

A Demonstration Program

We can do all this in a small Basic program to show that it all works. Let's start by following the suggested sequence of POKEs:

```
100 rem character set demo
110 poke 56576,4     (set 16K bank)
120 poke 53272,32    (set screen & ch set)
130 poke 648, 200    (tell Basic)
```

Next we wish to move the character set. . . but let's make this a visual thing. We'll print something on the screen first. This will initially appear as scrambled information, since the characters haven't been defined. As the characters are copied over from ROM, we'll see them take shape.

```
140 print chr$(147)    (clear screen)
150 for j=1 to 2
160 print "the quick brown fox jumped over
      a lazy dog's back 1234567890"
170 print chr$(18);
180 next j
```

Now let's copy the character set:

```
200 for j=0 to 2047
210 poke 56333, 127     (lockout interrupt)
220 poke 1, 51          (reveal chargen)
230 x=peek(j+53248)     (get 1/8 character)
240 poke 1, 55          (restore I/O)
250 poke 56333, 129     (free interrupt)
260 poke j+49152, x     (copy character part)
270 next j
```

One final touch. We have copied the existing character set. . . but we haven't defined anything new. Let's prove we can do it now by adding a slash to the letter "O" – this will make it look a little like the greek letter theta, or a sloppy number eight. "O" is the fifteenth letter of the alphabet, and each letter takes up eight bytes. So we can calculate that O starts at 49152 plus eight times 15. . . that gives 49272. Since we want to change the middle of the letter, not the top, we move down three more bytes and type:

```
280 poke 49275, 126
```

Check your program (mistakes can be fatal) and then RUN it. It's sufficiently slow that you can see the various characters being formed. The alphabetics come first, followed by the numerics; some time later, the reverse characters are defined.

Conclusion

There are lots of ways of setting up your own character set. This is just one way.

The first time is the hardest. . . once you've done it successfully, you'll be able to cut your own characters without problems.

Machine Code Version — **G.A. Campbell**
Toronto, Ont.

You've just seen how to do it in BASIC. Now let's try it in machine language.

The attached program disbles the interrupts, enables access

to the character generator (which simultaneously disables all I/O), and then copies the contents of the character generator ROM into RAM. The final activities are to turn I/O back on, allow interrupts, and tell the video controller to get the character definitions from RAM. The procedure is the same in BASIC, only the instructions change.

Once you have run this program, you can modify the character set by POKEing into the appropriate RAM. [Once again] The character definitions begin at memory location 49152 (hex $C000). Each character takes up 8 bytes of memory, beginning with the top row of dots for the character working down.

The offset for a given character is 8 times the "screen code" for that character plus the start address of the character base. Appendix E of the C64 User Guide lists the screen codes. The entire set takes 4K. The first K is the normal Upper case and graphics set. The second is the same, only in reverse field. Normal Upper/Lower case is in the third K, and the fourth K is same only reversed.

For example, if we want to change the "ampersand" ('&') into a square block in the middle of the character position, we would follow these steps:

1. Map out the POKEs to be used:

```
........  - 0
........  - 0
..xxxx..  - 60 (32+16+8+4)
..xxxx..  - 60
..xxxx..  - 60
..xxxx..  - 60
........  - 0
........  - 0
```

2. Look up the screen code (ampersand is 38)

3. Calculate it's address:

"&" loc = 49152 + 38*8 = 49456

4. Code the POKEs:

```
poke 49456, 0
poke 49457, 0
poke 49458, 60
poke 49459, 60
poke 49460, 60
poke 49461, 60
poke 49462, 0
poke 49463, 0
```

The Basic Loader below is followed by the machine code source listing for the same routine. The program is self modifying. While this is generally bad practice, it does make the program smaller and easier to type in.

BASIC Loader

```
6000 for j=832 to 881 : read x : ch=ch+x
6010 poke j, x : next
6020 if ch<>6113 then print "data error" : stop
6030 print "S"
6040 sys 832
6050 return
6060 data 120, 169,  51, 133,   1, 160
6070 data 224, 162,   0, 189,   0, 208
6080 data 157,   0, 192, 232, 208, 247
6090 data 238,  75,   3, 238,  78,   3
6100 data 204,  75,   3, 208, 236, 169
6110 data  55, 133,   1,  88, 169,  32
6120 data 141,  24, 208, 169,   4, 141
6130 data   0, 221, 169, 200, 141, 136
6140 data   2,  96
```

Editor's Note

The program above is Gord's machine language character mover modified slightly to reflect what Jim is doing in Basic. Gord's original version appears below which matches what Paul Higginbottom is doing in his Commodore 64 character editor this issue. If you like, the following lines can replace lines 6000–6090 of Paul's program to make the character transfer a little quicker. The above version could also be used, but this would mean some of the other variables in Paul's program would need changing.

Source Code Listing

```
100:                ; copy character set
110:                ; from rom to ram
120:                ; on the c64
                    ;
140: 0340           *       =   $0340
150: 0340           romctl  =   $01
160: 0340           basctl  =   $0288
170: 0340           chrram  =   $c000
180: 0340           chrset  =   $d000
190: 0340           vicctl  =   $d018
200: 0340           vicadr  =   $dd00
                    ;
220: 0340 78            sei             ;no interrupts
230: 0341 a9 33         lda  #51        ;disable i/o
240: 0343 85 01         sta  romctl
250: 0345 a0 e0         ldy  #$e0       ;limit for loop
260: 0347 a2 00         ldx  #0
270: 0349 bd 00 d0  loop    lda  chrset,x
280: 034c 9d 00 c0      sta  chrram,x
290: 034f  e8           inx
300: 0350 d0 f7         bne  loop
310: 0352 ee 4b 03      inc  loop+2     ;bump source
320: 0355 ee 4e 03      inc  loop+5     ;bump destination
330: 0358 cc  4b 03     cpy  loop+2
340: 035b d0 ec         bne  loop
                    ;
360: 035d a9 37         lda  #55        ;put i/o back
370: 035f  85 01        sta  romctl
380: 0361 58            cli             ;allow interrupts
390: 0362 a9 20         lda  #32
400: 0364 8d 18 d0      sta  vicctl     ;point at ram
410: 0367 a9 04         lda  #4
420: 0369 8d 00 dd      sta  vicadr     ;point vic2 at block3
430: 036c a9 c8         lda  #200
440: 036e 8d 88 02      sta  basctl     ;for basic
450: 0371 60            rts
```

Basic Loader: charam @ $3000, screen @ $0400

```
6000 for j=832 to 881 : read x : ch=ch+x
6010 poke j, x : next
6020 if ch<>5773 then print " data error " : stop
6030 print " S "
6040 sys 832
6050 return
6060 data 120, 169,  51, 133,   1, 160
6070 data 224, 162,   0, 189,   0, 208
6080 data 157,   0,  48, 232, 208, 247
6090 data 238,  75,   3, 238,  78,   3
6100 data 204,  75,   3, 208, 236, 169
6110 data  55, 133,   1,  88, 169,  29
6120 data 141,  24, 208, 169,   7, 141
6130 data   0, 221, 169,   4, 141, 136
6140 data   2,  96
```

Commodore 64 Programmable Character Editor

Paul Higginbottom
Toronto, Ont.

Once you've relocated the contents of the C64 character ROM into RAM, you'll probably want to start replacing the built–in characters with those of your own design. After all, isn't that why you moved the character set in the first place?

You could start POKEing the new character set locations (character base) with values that you've calculated by adding up bits, etc, but that's a lot of work. Besides, if you hit one byte wrong, you'll end up altering the wrong part of the target character, or you could miss the target completely!

This program eliminates the work. After all, what are computers for? First it moves the character ROM into RAM. Line 105 checks to see if this has been done so it only does it once. The new character set is placed at $3000. Your BASIC program is now limited to about 10K. Any bigger and BASIC text will start overwriting your characters and you'll be typing in cosmic hyrogliphics. When the new character set is finished, you can save it to disk and load it back later, possibly to another memory space if the VIC II chip has been moved.

First off, the program asks you to Select a character. This is mainly for convenience. If the character you have in mind resembles another character, you can select that character as a starting "pattern". For a "clean slate" simply hit Return and the character selected defaults to a space.

The character you selected will now be displayed on an 8x8 grid. Use the cursor keys to move around the grid. Cursor Home puts the cursor at the top left corner. To clear a dot or "pixel", hit space. To turn a pixel on, hit the "." (period) key. To abort, hit Clear Screen and the "Select?" prompt will return.

Once you're satisfied with the appearance of your new character, hit Return. The program will now ask which character you wish to substitute or "Use" for the new character. This means you must give up one character to make room for the new. You can use any character including graphics or reverse field. To replace a reverse field character, hit CTRL and RVS ON, then type your choice and hit Return. The memory that holds the pattern for this character will now be overwritten with the new pattern. Whenever this key is typed, you'll see the results of that replacement until it is modified again, or the machine is reset. If you've inadvertently hit Return to enter the Use? prompt, hit Clear Screen and you will be allowed to do further editing.

Finally, when you're through entering all the special characters you need, get back to the "Select?" prompt and hit Home. After entering a filename, the program will store the contents of the RAM based character set in a sequential disk file (if you have a disk). Only the non–reverse characters are stored. If you want the whole set filed, change the "2047" in line 9010 to 4095.

There is no provision in this program to retrieve these files, although it wouldn't be hard to add. . . simply complement the save routine at 9000. But a function like this would normally be found in another program that makes use of your new characters. A utility for providing new character sets was the main purpose of this program.

Editor's Note

Paul's program moves the ROM character set to RAM with a BASIC subroutine. While this is happening, your 64 will be pretty quiet. Be patient though. . . the first prompt will appear shortly. However if you're the impatient type like me, replace the subroutine at line 6000 with Gord Campbell's machine language version. See the article on Moving the C64 Character Set this issue.

```
100 vic=13*4096 : c=3*4096 : sc=1024
105 if peek(vic+24)<>29 then gosub 6000
110 for i=0 to 7 : m(i)=2↑(7-i) : h(i)=255-m(i) : next
120 z=c+255*8 : cl=sc+3*40+15 : dc=255
200 gosub 500
210 for i=0 to 7 : p=peek(l+i) : poke z+i, p
220 for j=0 to 7 : a$="."
230 if p and m(j) then a$="Q"
240 print a$; : next : print : next : x=0 : y=0
250 print "■"; : poke cl, dc
260 if y then for i=1 to y : print : next
270 if x then print spc(x);
280 p=sc+x+y*40 : q=peek(p) : r=q
290 r=(notrand128) or (not128andr)
300 poke p, r
310 for i=1 to 30
315 get a$ : if a$="" then next : goto 290
320 poke p, q
330 if a$="■" then x=(x+1)and7 : goto 250
340 if a$="[crsr lft]" then x=(x-1)and7 : goto 250
350 if a$="■" then y=(y+1)and7 : goto 250
360 if a$="■" then y=(y-1)and7 : goto 250
370 if a$=" " then poke p, 46 : t=z+y : poke t, peek(t) and h(x) : x=(x+1) and 7 : goto 250
380 if a$="." then poke p, 81 : t=z+y : poke t, peek(t) or m(x) : x=(x+1) and 7 : goto 250
390 if a$="■" then 200
395 if a$="■" then x=0 : y=0 : goto 250
400 if a$<>chr$(13) then 250
410 print "■■■■■■■■■use?";
411 rem important!! above is crsr home, 9 cursor downs, u-s-e-?
415 gosub 600 : if a$="■" then print : print "■      " : goto 250
417 print a$; : if a$<>chr$(13) then 410
420 p=peek(1388) : l=c+p*8
440 for i=0 to 7 : v(i)=0
450 for j=0 to 7 : if peek(sc+i*40+j)=81 then v(i)=v(i)+m(j)
460 next j, i
470 for i=0 to 7 : poke l+i, v(i) : next
480 print "■" : poke 1065, p : goto 200
500 print "■←-Select?■";
505 gosub 600 : print a$"■"; : if a$="■" then gosub 9000 : goto 500
506 if a$<>chr$(13) then 505
510 p=peek(sc) : l=c+p*8 : print "■"; : return
600 get a$ : if a$="" then 600
610 return
6000 cia=13*4096 + 12*256
6010 poke cia+13, 127
6020 poke 1, peek(1) and 251
6030 oc=13*4096
6040 for i=0 to 256*8-1
6050 poke c+i, peek(oc+i) : next
6060 poke 1, peek(1) or 4
6070 poke cia+13, 255
6080 poke vic+24, 29 : return
9000 input "■name";f$ : open 2, 8, 2, "0:"+f$+",p,w"
9010 for i=c to c+2047 : print#2, chr$(peek(i)); : next : close 2 : return
```

Commodore 64 To 8023P Hi – Res Screen Dumper

David Berezowski
Toronto, Ont.

If you're using your Commodore 64 to make high resolution pictures, graphs, charts, or otherwise, this program will allow you to dump screen contents to a Commodore 8023P dot matrix printer.

If you don't have a Commodore 64, the program will also work on PET/CBMs. This way, if you get your hands on a disk file of a C64 hi–res screen, you too can obtain a hard copy repro.

The program is self documenting, and although you need not type in all the instructions, I suggest you do as this will avoid any possible snafus later.

The machine language portion is set up first. It starts at $1F00 which is just 255 bytes below $2000. The machine code assumes that the C64 high res screen is (or was) set up at $2000. If you are going to be loading high res files from disk that were stored from a high res screen at a different location, you may want to replace line 370 with a GOSUB to a short loader subroutine:

```
370 gosub 1000

1000 open 8, 8, 8, fl$
1010 get#8, a$, a$ : rem discard start addr
1020 j=0 : sc=8192
1030 get#8, a$
1040 poke sc+j, asc(a$+chr$(0))
1050 j=j+1
1060 if st then close 8 : return
1070 goto 1030
```

Editor's Note:

Commodore's new 6400 and 1527 printers also have an 8x8 dot matrix like the 8023P. The program has not been tested on them but it will probably work with little or no modification. The only possible trouble spots I can see might lie in the OPEN commands to the printer in lines 380, 390, and 410. The secondary addresses may change for things like line spacing and programmable character accept.

```
100 if c then 380
110 q$=chr$(34)
120 print "** c64 hi-res to 8023p dumper **"
130 print "(david berezowski - aug/82)"
140 print "this program dumps the hi-res"
150 print "screen of the commodore-64 to"
160 print "a 8023p printer."                    : print
170 print "this program assumes that:"
180 print "1) the hi-res page is stored"
190 print "as a program file on disk,"
200 print "with a load address of $2000."       : print
210 print "      or"                            : print
220 print "2) the hi-res page resides"
230 print "in memory from $2000 to $3FFF."      : print
240 print "this program will work on any
```

```
250 print "pet, cbm, vic, or c64 system."  : print
260 gosub 430
270 print "hit any key to continue"
280 get a$ : if a$="" then 280
290 print
300 input "'e'xit or dump from 'd'isk or 'm'emory";x$
310 print
320 if x$<>"d" and x$<>"e" and x$<>"m" then 300
330 if x$="m" then 380
340 if x$="e" then end
350 input "enter file name";fl$
360 print : print "loading file..."
370 c=1 : load fl$, 8, 1
380 open  6, 4,  6 : print#6, chr$(8) : close 6
390 open 17, 4, 17 : open 18, 4, 18
400 sys 7936       : rem $1f00
410 open  6, 4,  6 : print#6, chr$(12): close 6
420 goto 290
430 ch=0 : for j=7936 to 8182 : read x : poke j, x
440 ch=ch+x : next : if ch<>27502 then print "data error" : stop
450 return
460 data 165,   2,  72, 165,   3,  72, 162,  25
470 data 142, 247,  31, 162,   0, 142, 253,  31
480 data 142, 254,  31, 162,  40, 142, 248,  31
490 data 162,   0, 142, 251,  31, 142, 252,  31
500 data 162,   8, 142, 249,  31, 162,  17,  32
510 data 201, 255, 162,   0, 142, 255,  31, 162
520 data   8, 142, 250,  31, 162,   0, 134,   2
530 data 162,  32, 134,   3, 165,   2,  24, 174
540 data 250,  31, 125, 238,  31, 144,   2, 230
550 data   3,  24, 109, 251,  31, 144,   2, 230
560 data   3,  24, 109, 253,  31, 144,   2, 230
570 data   3, 133,   2,  24, 165,   3, 109, 252
580 data  31, 109, 254,  31, 133,   3, 162,   0
590 data 161,   2, 174, 249,  31,  61, 222,  31
600 data 240,  13, 173, 255,  31,  24, 174, 250
610 data  31, 125, 230,  31, 141, 255,  31, 206
620 data 250,  31, 208, 176,  56, 169, 255, 237
630 data 255,  31,  32, 210, 255, 206, 249,  31
640 data 208, 152, 173, 251,  31,  24, 105,   8
650 data 144,   3, 238, 252,  31, 141, 251,  31
660 data 206, 248,  31, 240,   3,  76,  32,  31
670 data 162,  18,  32, 201, 255, 169,  13,  32
680 data 210, 255,  32, 204, 255,  32, 228, 255
690 data 201,   3, 240,  25, 173, 253,  31,  24
700 data 105,  64, 144,   3, 238, 254,  31, 141
710 data 253,  31, 238, 254,  31, 206, 247,  31
720 data 240,   3,  76,  19,  31,  32, 231, 255
730 data 104, 133,   3, 104, 133,   2,  96,   1
740 data   2,   4,   8,  16,  32,  64, 128,   1
750 data   2,   4,   8,  16,  32,  64, 128,   7
760 data   6,   5,   4,   3,   2,   1,   0
```

VIC 20 / Commodore 64
BASIC Line Labeling

The program that follows was originally presented in The Transactor by J Hoogstraat of Calgary, Alberta, back in 1979. Shortly afterwards, Charles A. McCarthy of St. Paul, Minnesota updated the routine to eliminate some bugs and make it relocatable. Since then, a BASIC 4.0 version has been published and other changes have been made. Utilities come and go, but this one has been so popular amongst Transactor readers, that versions for the VIC 20 and Commodore 64 were felt to be in order.

In Commodore BASIC, a GOTO, GOSUB, IF THEN/GOTO, or ON GOTO/GOSUB statement must be followed by a line number, a most unnatural way to transfer execution because a line number is not a very descriptive about the target code. Furthermore, if the program is re–numbered using a programmers aid (such as Dave Hook's Tiny–Aid for the 64, this issue), the line numbers change and the programmer must remember a whole new set of line numbers to call frequently used subroutines and entry points.

With this utility, line numbers can be replaced by "line labels". For example:

GOTO 100	might become	GOTO &START
GOSUB 50000		GOSUB &KEYBOARD
IF ST THEN 10000		IF ST THEN &CLOSEFILES

Unlike a line number, a label offers the programmer a means of executing code by referencing a meaningful target. Also, when a program is re–numbered, the lines that use labels as the target are unaffected. The programmer can continue calling subroutines, etc, using the old line labels without regard for the new line numbers.

Other languages such as COMAL, PASCAL, and Waterloo BASIC discourage the use of line numbers altogether. Line labels are used almost without exception. However, programs written in these languages are only transportable to machines that have same language installed. This utility gives Commodore BASIC users one of the advantages of a structured language while maintaining the programs "machine independence."

But you say, "that means this utility must be included with the program before it will run on another machine". Although it certainly can be included within a program, is was meant more as a development tool. When a program reaches its final version, a "search and replace" utility would be used to substitute the labels back to their corresponding line numbers.

Utility Utilization

Probably the best way to demonstrate line labels is with two examples; one with labels, the other without. Some preliminary explanation:

First, the utility does not require exclusive use of labels. Line numbers and line labels can be used at will.

Second, all labels must be preceded by an ampersand (&). This routine links itself to BASIC via the CHRGET subroutine in low RAM. (see your memory map) When CHRGET is called from ROM, a character is returned from BASIC text space that somehow gets manipulated by BASIC. If this character is an "&", and if it follows a GOTO, GOSUB, etc, the utility takes over. Originally the label indicator was a "#" sign but was changed due to interference with commands such as GET#, DCLOSE#, RECORD#, etc.

Next, destinations can be labeled in a number of ways. The label can be on a line by itself or on the first line of code followed by a colon (see example). Labels must be a single word (ie no spaces, commas, or colons) but can be any length up to the maximum characters you can fit on a line. The label may even include embedded BASIC keywords. But when referencing a label, the entire label must be used each time. Unlike variable names in CBM BASIC, all the characters must match. If not, ?Undef'd Statement Error. Keep labels short in length if you intend to use multiple statement lines, and also to avoid exact spelling mismatches.

Finally, the Commodore 64 version works with or without the RTC C64–Link which also patches into BASIC. If you

have another cartridge installed that interferes with BASIC (eg. VSP), chances are the utility will malfunction. The VIC 20 version also works with V-Link, but it has only been tested with 8K of expansion RAM. However, there should be no problem with other configurations. If there is, please inform us.

Technical Info

As mentioned earlier, this utility modifies the CHRGET subroutine in low RAM. If another utility is in place that also wedges into BASIC via CHRGET, it will be disabled when this one is POKEd in. Likewise, if another routine that modifies CHRGET is brought in at some later time, the labeling utility will be disabled by the new routine.

A more contemporary method to link this routine is through the New Basic Code Link (see memory map). This link is an indirect jump vector through which the Basic interpreter examines code. By re-routing it, like V-Link, new code can be added, special characters can be scanned for, etc. The labeling utility is actually linked "the old way". Although this is not as efficient as the Basic Code Link method, it allows the routine to resemble more closely the previous versions.

Some utilities link into BASIC via CHRGOT. Dave Hook's VIC 20/C64 Tiny-Aid works this way. Since the two utilities are modifying different parts of the CHRGET subroutine, they allow for a very comfortable co-existence. Such a combination offers the luxury of both line labels AND additional editing commands.

Note: If you search and replace the labels at some later time using Tiny-Aid, all of the occurences will get replaced including the one marking the entry point. Generally it's best to change this spot before the search and replace. You'll either need to delete the label or put a REM behind it.

At the very beginning of the routine, program or direct mode is tested. The utility will abort if a GOTO &Label is issued directly from the keyboard.

The code is entirely relocatable. This means that it will work no matter where it is set up in RAM. Since there are no internal JMPs, a simple BASIC relocating loader was chosen to establish the routine, which puts it directly underneath the Limit of Memory pointer. This also makes the utility compatible with other utilities that get placed in high RAM.

Summary

Referencing BASIC lines by a label as opposed to a number will make programming life a lot easier. Development time is reduced and silly mistakes are albeit eliminated, especially in larger programs. Once you've written a program using the Basic Line Labeling utility, you'll wonder how you ever got by without it!

Example Without Labels

```
100 for i = 1 to 3
110 on i gosub 160, 190, 220
120 next
130 :
140 goto 250
150 :
160 rem
170 print " subroutine 1 " :return
180 :
190 print " second subroutine
200 return
210 :
220 print " # three
230 return
240 :
250 a=0
260 a=a+1 : print a
270 if a<10 then 260
280 :
290 goto100
```

Same Example With Labels

```
100 for i = 1 to 3
110 on i gosub &sub1, &two, &onemore
120 next
130 :
140 goto &alldone
150 :
160 &sub1 : rem label on own line
170 print " subroutine 1 " :return
180 :
190 &two : print " second label on line with code
200 return
210 :
220 &onemore   : print " # three has embedded basic + extra spcs
230 return
240 :
250 &alldone:a=0
260 &notyet : a=a+1 : print a
270 if a<10 then &notyet
280 :
290 goto100
```

```
900 rem c64 basic labeling utility
910 ad=peek(55)+peek(56)*256-205
920 poke 55, ad-int(ad/256)*256
930 poke 56, int(ad/256)
940 for j=ad to ad+205-1
950 read x : poke j, x
960 ch=ch+x
970 next
980 if ch<>25202 then print "data error" :stop
990 sys ad : print "sys";ad;"to re-enable"
1000 clr
1010 data 169,  76, 133, 115, 165,  55
1020 data  24, 105,  21, 133, 116, 165
1030 data  56, 105,   0, 133, 117,  96
1040 data  76, 227, 168, 230, 122, 208
1050 data   2, 230, 123, 138,  72, 162
1060 data 255, 228,  58, 240,   6, 161
1070 data 123, 201,  38, 240,   5, 104
1080 data 170,  76, 121,   0, 104, 170
1090 data 104, 201, 221, 208,   4,  72
1100 data  76, 121,   0, 201,  54, 240
1110 data  29, 201, 210, 240,  25, 201
1120 data 159, 240,  29, 201, 130, 240
1130 data  28,  72, 201,  97, 208, 121
1140 data 104, 104,  32, 115,   0, 201
1150 data  44, 208, 249,  76,  87, 169
1160 data 165,  97, 208,   4, 104,  76
1170 data  59, 169,  56, 176,   1,  24
1180 data 104,   8, 165,  43, 166,  44
1190 data 160,   0, 240,   4, 160,   0
1200 data 177,  95,  24, 133,  95, 105
1210 data   4, 133,  93, 138, 133,  96
1220 data 105,   0, 133,  94, 200, 177
1230 data  95, 240, 139, 170, 136, 177
1240 data  93, 240,  15, 201,  58, 240
1250 data  11, 201,  32, 240,   7, 209
1260 data 122, 208, 215, 200, 208, 237
1270 data 177, 122, 240,  12, 201,  44
1280 data 240,   8, 201,  58, 240,   4
1290 data 201,  32, 208, 196,  40, 176
1300 data  21, 165, 123,  72, 165, 122
1310 data  72, 165,  58,  72, 165,  57
1320 data  72, 169, 141,  72, 169, 167
1330 data  72, 169, 173,  72,  32, 197
1340 data 168,  32, 248, 168,  76, 121
1350 data   0
```

```
900 rem vic 20 basic labeling utility
910 ad=peek(55)+peek(56)*256-205
920 poke 55, ad-int(ad/256)*256
930 poke 56, int(ad/256)
940 for j=ad to ad+205-1
950 read x : poke j, x
960 ch=ch+x
970 next
980 if ch<>25331 then print "data error" :stop
990 sys ad : print "sys";ad;"to re-enable"
1000 clr
1010 data 169,  76, 133, 115, 165,  55
1020 data  24, 105,  21, 133, 116, 165
1030 data  56, 105,   0, 133, 117,  96
1040 data  76, 227, 200, 230, 122, 208
1050 data   2, 230, 123, 138,  72, 162
1060 data 255, 228,  58, 240,   6, 161
1070 data 123, 201,  38, 240,   5, 104
1080 data 170,  76, 121,   0, 104, 170
1090 data 104, 201, 131, 208,   4,  72
1100 data  76, 121,   0, 201,  54, 240
1110 data  29, 201, 237, 240,  25, 201
1120 data 159, 240,  29, 201, 130, 240
1130 data  28,  72, 201,  97, 208, 121
1140 data 104, 104,  32, 115,   0, 201
1150 data  44, 208, 249,  76,  87, 201
1160 data 165,  97, 208,   4, 104,  76
1170 data  59, 201,  56, 176,   1,  24
1180 data 104,   8, 165,  43, 166,  44
1190 data 160,   0, 240,   4, 160,   0
1200 data 177,  95,  24, 133,  95, 105
1210 data   4, 133,  93, 138, 133,  96
1220 data 105,   0, 133,  94, 200, 177
1230 data  95, 240, 139, 170, 136, 177
1240 data  93, 240,  15, 201,  58, 240
1250 data  11, 201,  32, 240,   7, 209
1260 data 122, 208, 215, 200, 208, 237
1270 data 177, 122, 240,  12, 201,  44
1280 data 240,   8, 201,  58, 240,   4
1290 data 201,  32, 208, 196,  40, 176
1300 data  21, 165, 123,  72, 165, 122
1310 data  72, 165,  58,  72, 165,  57
1320 data  72, 169, 141,  72, 169, 199
1330 data  72, 169, 173,  72,  32, 197
1340 data 200,  32, 248, 200,  76, 121
1350 data   0
```

Source Code Listing

The source code below shows a start address of $033c. This is just a dummy start address and is only necessary to satisfy the assembler (remember, the machine codes are POKEd in with a relocator). The routine is actually too big to fit at $033c so watch that your assembler doesn't put it there or it will clobber something for sure.

The listing shown is for the Commodore 64 version. The VIC 20 source is identical except for a few addresses (as you might notice by comparing the two loaders). Any addresses that change in the VIC version are shown in brackets.

```
100: 033c                *  =   $033c      ;dummy, do not assmbl here

120: 033c a9 4c          lda  #$4c        ;sys here to link (AD in loader)
130: 033e 85 73          sta  $0073
140: 0340 a5 37          lda  $0037
150: 0342 18             clc
160: 0343 69 15          adc  #21
170: 0345 85 74          sta  $0074
180: 0347 a5 38          lda  $0038
190: 0349 69 00          adc  #00
200: 034b 85 75          sta  $0075
210: 034d 60             rts
220: 034e 4c e3 a8 undefd jmp  $a8e3     ;undef'd statement error($c8e3)
230: 0351 e6 7a    entry  inc  $7a       ;perform TXTPTR inc
240: 0353 d0 02           bne  nohinc    ;for CHRGET
250: 0355 e6 7b           inc  $7b
                   ;
270: 0357 8a       nohinc txa            ;save .X
280: 0358 48              pha            ;on stack
290: 0359 a2 ff           ldx  #$ff      ;test immediate mode
300: 035b e4 3a           cpx  $3a
310: 035d f0  06          beq  exit      ;if so, exit
320: 035f  a1 7b          lda  ($7b,x)   ;no, check for &
330: 0361 c9 26           cmp #$26
340: 0363 f0  05          beq  chklab    ;found, go test
                   ;
360: 0365 68       exit   pla            ;no, restore .X
370: 0366 aa              tax
380: 0367 4c 79 00        jmp  $0079     ;return to basic
                   ;
400: 036a 68       chklab pla            ;restore .X
410: 036b aa              tax
420: 036c 68              pla            ;pull lo byt of calling addr
430: 036d c9 dd           cmp #$dd       ;C64 Link scan (#$83)
440: 036f  d0 04          bne  chkthn    ;no, go on
450: 0371 48              pha            ;yes, ignore
460: 0372 4c 79 00        jmp  $0079
470: 0375 c9 36    chkthn cmp #$36       ;basic 'then'?
480: 0377 f0  1d          beq  chkcond   ;yes, chk condition
490: 0379 c9 d2           cmp #$d2       ;64-Link 'then'? (#$ed)
500: 037b f0  19          beq  chkcond
510: 037d c9 9f           cmp #$9f       ;goto?
520: 037f  f0  1d         beq  entr1     ;yes, go find transfer
```

```
530: 0381 c9 82          cmp #$82      ;gosub?
540: 0383 f0  1c         beq  entr0    ;yes, go find
550: 0385 48             pha           ;lo byte to stack in case
560: 0386 c9 61          cmp #$61      ;nothing to do
570: 0388 d0 79          bne  exit2    ;no, skip statement
580: 038a 68             pla           ;yes, do 'on'
590: 038b 68             pla           ;pull rtn addr
                         ;
610: 038c 20 73 00 fndcom jsr  $0073   ;advance TXTPTR
620: 038f  c9 2c         cmp #$2c      ;tocomma
630: 0391 d0 f9          bne  fndcom   ;following label
640: 0393 4c 57 a9       jmp  $a957    ;cont with on
                         ;
660: 0396 a5 61  chkcond lda  $61      ;chk condition for if
670: 0398 d0 04          bne  entr1    ;true, go on
680: 039a 68             pla           ;false, pull rest of addr
690: 039b 4c 3b a9       jmp  $a93b    ;and skip line
700: 039e 38     entr1   sec           ;carry flag set for goto/then
710: 039f  b0 01         bcs  entrgo
720: 03a1 18     entr0   clc           ;carry clr to remember gosub
                         ;
740: 03a2 68     entrgo  pla           ;pull rest of addr
750: 03a3 08             php           ;to save carry flag
760: 03a4 a5 2b  flabel  lda  $2b      ;init basic txtptr to
770: 03a6 a6 2c          ldx  $2c      ;search for line starting
780: 03a8 a0 00          ldy  #$00     ;with &label
790: 03aa f0  04         beq  srchln   ;forced branch
                         ;
810: 03ac a0 00  nxstat  ldy  #$00     ;get pointer to
820: 03ae b1 5f          lda  ($5f),y  ;present line in $5f
                         ;
840: 03b0 18     srchln  clc
850: 03b1 85 5f          sta  $5f
860: 03b3 69 04          adc  #$04
870: 03b5 85 5d          sta  $5d
880: 03b7 8a             txa
890: 03b8 85 60          sta  $60
900: 03ba 69 00          adc  #$00
910: 03bc 85 5e          sta  $5e
920: 03be c8             iny           ;.Y = 1
930: 03bf  b1 5f         lda  ($5f),y  ;chk for null link, endprog
940: 03c1 f0  8b         beq  undefd   ;if so, label not found
950: 03c3 aa             tax
960: 03c4 88             dey
                         ;
980: 03c5 b1 5d  match   lda  ($5d),y  ;test for &label terminator
990: 03c7 f0  0f         beq  fndtrm   ;null?, end of line
1000: 03c9 c9 3a         cmp #$3a      ;colon, end of statement
1010: 03cb f0  0b        beq  fndtrm
1020: 03cd c9 20         cmp #$20      ;space?
1030: 03cf  f0  07       beq  fndtrm   ;no terminator, test against
1040: 03d1 d1 7a         cmp ($7a),y   ;given &label - to next statement
1050: 03d3 d0 d7         bne  nxstat   ;if not this one. Match so far
```

```
1060: 03d5 c8                   iny         ;test next character
1070: 03d6 d0 ed                bne  match  ;branch always
                        ;
1090: 03d8 b1 7a    fndtrm      lda  ($7a),y ;found termin. this line
1100: 03da f0  0c               beq  trnsfrex ;test given &label for termin.
1110: 03dc c9 2c                cmp #$2c    ;termins. are null, comma,
1120: 03de f0  08               beq  trnsfrex
1130: 03e0 c9 3a                cmp #$3a    ;colon,
1140: 03e2 f0  04               beq  trnsfrex
1150: 03e4 c9 20                cmp #$20    ;space
1160: 03e6 d0 c4                bne  nxstat ;no termin, try next line
                        ;
1180: 03e8 28       trnsfrex    plp         ;recall then/goto vs. gosub
1190: 03e9 b0 15                bcs  nosub  ;set?, not a sub.
1200: 03eb a5 7b                lda  $7b    ;clr?, put return data on stack
1210: 03ed 48                   pha
1220: 03ee a5 7a                lda  $7a
1230: 03f0  48                  pha
1240: 03f1  a5 3a               lda  $3a
1250: 03f3  48                  pha
1260: 03f4  a5 39               lda  $39
1270: 03f6  48                  pha
1280: 03f7  a9 8d               lda  #$8d
1290: 03f9  48                  pha
1300: 03fa  a9 a7               lda  #$a7   ;(#$c7)
1310: 03fc  48                  pha
1320: 03fd  a9 ad               lda  #$ad
1330: 03ff  48                  pha
                        ;
1350: 0400 20 c5 a8 nosub       jsr  $a8c5  ;set new execution($c8c5)
                        ;
1370: 0403 20 f8  a8 exit2      jsr  $a8f8  ;skip rest of statement($c8f8)
1380: 0406 4c 79 00             jmp  $0079  ;return to chrgot
```

Tiny–Aid For The Commodore 64

David A. Hook
Barrie, Ont.

Introduction

Readers of the Transactor (Volume 4, Issue 1) and COMPUTE! (#31 December 1982) may have noticed that they contained a VIC version of Jim Butterfield's Tiny Aid for PET. In answer to popular demand, but mainly because I finally got my Commodore 64, here is the further adaptation for our favourite company's newest wunderkind.

Although I realize memory size is not a serious problem on the C64, there are still only a few of the "complete 4K" Basic–Aid features that I use frequently. In keeping with the original spirit of Tiny Aid, you won't find much different in this package.

Never let it be said that we are standing still, however. You will note that this one does have an APPEND command, which works with either tape or disk. (It is not a MERGE, so you will have to do your own housekeeping with proper staging of line number sequences).

For your program editing satisfaction, I present to you C64 TINY AID.

Features

C64 TINY AID is a machine language program which consumes about 1177 bytes of your RAM memory. After you have loaded the program, type 'RUN' and hit <RETURN>. The program automatically repacks itself into high memory. It goes in at the current top of memory, and will peacefully co–exist with any other programs above it. The appropriate memory pointers are set so that Basic will not clobber it. C64 TINY AID is now alive.

Once activated, the six commands become attached to Basic. They will function only in "direct" mode, i.e. don't include them in a program.

(1) NUMBER 1000,5 <RETURN>
NUMBER 100,10

Renumbers a Basic program with a given starting line number and given increment between line numbers. The maximum increment is 255.

All line number references after GOTO, THEN, GOSUB and RUN are automatically corrected. A display of these referenced lines is presented on the screen as the command does its thing. If you have a GOTO which refers to a non–existent line number in your program, then it is changed to 65535. This is an illegal line number, and must be corrected before the Basic program is used.

(2) DELETE 100–200 <RETURN>
DELETE – 1500
DELETE 5199 –

Deletes a range of lines from a Basic program. Uses the same syntax as the LIST command, so any line–range may be specified for removal. DELETE with no range will perform like a NEW command, so be careful.

(3) FIND /PRINT/ <RETURN>
FIND /A$/, 150–670
FIND "PRINT", 2000–
FIND *62050*, –3110

Will locate any occurences of the characters between the special marks called "delimiters". Almost any character may be used to identify the start and the end of the "string" to be found, so long as both are the same. The first example will find all the PRINT instructions in the program.

This "search string" is tokenized by the routine, which means you need to be careful if you are looking for a string

of text which contains a Basic "keyword". For example, you could be looking for the word "PRINT" as part of a text message, ie. in quotes. The third example above shows how to do this. Sometimes those keywords can hide on you, so it may be worthwhile to search using both kinds of start/end markers.

You can perform a search that is limited to a certain line range. Just specify the starting and ending line numbers as you would with a LIST command. Three examples above demonstrate this. Please note that a comma must separate the end-marker from the line range.

All lines containing the string are printed to the screen. If a line has more than one of them, each occurence will cause a repetition of that line on the screen.

(4) CHANGE -PRINT-PRINT#4,- <RETURN>
CHANGE /ABC/XYZ/, 6000-
CHANGE /DS$/D1$/, -5000
CHANGE "BOB"CAROL"

Using the same syntax as FIND, you may change any string to any other string in a Basic program. This command is very powerful, and was not part of the early versions of Basic-Aid or Toolkit. I use this one quite often.

As before, you may indicate a line-range. As the changes are made, the revised lines are displayed on the screen.

Watch out for the difference between Basic keywords and strings of text within quotes. You may use the quote characters to differentiate, as with FIND.

The first example shows how to change all the PRINT statements to address them to another device, perhaps a printer. Notice that the comma at the end of the "PRINT#" is inside the end marker. This will result in a SYNTAX ERROR if the PRINT was being used to do a blank line, so all such lines need to be changed back manually. It's your choice as to which works best with your programming style.

I always do a FIND before attempting the CHANGE, just to see the potential mess I'm about to create.

(5) APPEND "file name", 1 <RETURN>
APPEND "1:file name", 8

This is the new kid on the block, as mentioned above. Either tape or disk may be used as the source. The second example refers to drive #1 of a dual drive system. Its function is to append a Basic program to the end of the one currently in memory.

Several cautions are in order here. First, this is not a true MERGE, as the new program just attaches itself to the end of the one already there. Secondly, no check is made to see if the one in memory is "Basic-only". Thirdly, no check is made on the incoming one to see if it is Basic either. Finally, it assumes you are big boys and girls and won't attempt to APPEND programs that have lower line numbers than the resident program. All bets are off if you do these naughty things!

After all these caveats, what is it good for? You may have a series of subroutines stored on disk or tape which are used frequently (a bullet-proof input routine or a number formatting routine are two examples). Rather than having to LOAD them in first, you can bring them in after you've got underway with your programming. It was considered to be worthwhile by the person who asked me to include it in this version.

(6) KILL <RETURN>

This command disables C64 TINY AID and its associated commands. A syntax error will be the result if any of the above commands are now tried.

Since the routine is safe from interference from Basic, you may leave it active for as long as your machine stays on. It is possible that C64 TINY AID may interfere with other programs that modify Basic's internal 'CHRGOT' routine. The KILL command allows you to avoid this conflict.

Procedure

The C64 contains no internal machine language monitor, which is the most efficient way to enter this program. There are several options for you, depending on the state of your program library:

(1) If you have access to an Upgrade (2.0) or Basic 4.0 PET/CBM, it has an internal ML monitor. This will work quite well.

(2) Use your C64, but you must have a machine language monitor or something that does the same job:

-Jim Butterfield's SUPERMON64.V1 (Compute#32, January 1983).
-Jim Butterfield's TINY PEEKER/POKER (same article)

which I have (very slightly) modified in this article.

(3) The way of the Post Office. Send $3, a blank cassette or 1541/2031/4040 diskette in a stamped, self-addressed mailer to me at:

58 Steel Street
BARRIE, Ontario, CANADA
L4M 2E9

Be sure it's packaged securely. Diskettes will be returned in DOS 2.0 format. If you have a 1540/1541/2031 LP/or an old 2040 (which still has DOS 1.0) you should not SAVE or WRITE to the diskette you get from me. Prepare a diskette that you formatted on your own drive, LOAD the program into memory, insert your other disk and SAVE it. When you've have a VERIFYed copy, re-format the original disk for your own re-use. This caution is worthwhile because the "low-profile" drives are not "WRITE-compatible" with the other IEEE drives from the PET/CBM equipment. You should be able to "READ" the SAVEd programs OK.

By the way, I cannot use U.S. or other foreign postage to return your tape/disk. You would be amazed to learn the number of people who just followed my "orders" with the VIC 20 TINY AID blindly. Can anyone use $150 worth of American stamps, some in rather shabby condition?

If you're still with me, you are about to undertake the entry of about 3300 characters worth of hexadecimal numbers. As well as the digits from zero to nine, you will need the alphabetic characters from A-F to represent numbers from ten to fifteen. These characters, and three instructions, will be all that are used to enter our program. As I've often said before, just follow the instructions—it won't be the first computer-related thing you may not fully understand! But you can make it work without totally knowing the effect.

The alternative, a Basic loader with DATA statements, would be four times as much typing and would require as much "messing about" in the end anyway.

From this point on, we'll take different paths. I hope we all can get back together for the finale:

A. With a PET/CBM:

Start with a freshly-powered-up machine. We first need to move the Start-of-Basic Pointer to match the Commodore 64's. Type in the following instructions, (without using line numbers):

POKE 2048, 0 : POKE 41, 8 : NEW <RETURN>

Now type in the Basic program listing that appears with this article. Much of it represents internal documentation on how the commands work, so don't leave it out. The information will fit on the screen without losing any by scrolling, so type it in exactly, and don't add anything extra either.

Perform a normal Basic SAVE to your tape or disk, and VERIFY it too. Call it "C64 AID.BAS". Type 'NEW' to clear the program. That was pretty easy stuff.

We need to enter the ML monitor now, so type in: SYS2048 and hit <RETURN>. There will be a display on the screen, which you may ignore if you don't understand monitors. Below this you can see the cursor flashing next to a period character ("."). The period is like a prompt character in the ML scheme of things. Starting at the current cursor position, enter:

.M 0980 09F8 <RETURN>

Sixteen lines of stuff should appear on the screen, much like the "memory-dump" which accompanies this article. A four-*digit* quantity called an "address" leads off a line, and eight pairs of two-*digit* values appear alongside.

Look at the tables of values in the article. The first "block" has the address "0980", which matches the first address just above. The last row of this same table shows "09F8". (This same process will be repeated for each of the eleven tables given, ie. using the first and last "addresses" of each table).

You need to enter the values given, typing right over the values shown on the screen. Hit <RETURN> at the end of each "line" or else nothing will get remembered. Double check the values you've typed. It's not easy to find an error later on.

Look at the next block of values. Type in the start/end addresses to display:

.M 0A00 0A78 <RETURN>

Continue until all eleven tables' worth have been entered.

Now we need to save this stuff before anything goes wrong. To do this, we need the ML Monitor SAVE command as shown below:

.S "C64 AID.ML",01,0980,0F00 <RETURN> (tape)
.S "0:C64 AID.ML",08,0980,0F00 <RETURN> (disk)

Mount a tape or disk, and follow the instructions. Save a second copy, for safety. UNDER NO CONDITIONS SHOULD

YOU SAVE EVEN ONE EXTRA BYTE!

Exit the ML monitor, with:

.X <RETURN>

Rewind the tape, if applicable, and VERIFY the program normally before going any further.

Go to the section titled Check–Out, so that the ML part can be verified before we go any further. Come back here when the checksum is OK.

Type 'NEW', then LOAD the "C64 AID.BAS". Follow this immediately with a LOAD of "C64 AID.ML". This sets the Basic pointers properly so a normal SAVE command will get both pieces properly. SAVE this as "C64 AID.REL" and VERIFY it too.

Now proceed to the section on Final Checking.

B. C64 with Supermon64:

Again, it's best to start with a machine that's been turned off and back on again. LOAD and RUN the Supermon64 program first. You must type "X" (and <RETURN>) to exit the monitor for now, and then type 'NEW' to erase Supermon.

Refer back to the instructions in Part A, starting AFTER the line with the POKE statements. You can follow these directions all the way down to the paragraph starting: "Type 'NEW', then LOAD. . .". Since the C64 does a little trick called "program relocation", you need to place the ML part back into the right spot.

Do the Basic LOAD as given above for "C64 AID.BAS". Then you need to enter:

LOAD "C64 AID.ML",1,1 <RETURN> (tape)
LOAD "C64 AID.ML",8,1 <RETURN> (disk)

Now you may do the normal Basic SAVE as "C64 AID.REL" and VERIFY it. The Basic and machine language have been linked together and will now LOAD as one piece.

Now proceed to the section on Final Checking.

C. C64 with no ML monitor:

Let's start afresh by turning off the C64 and turning it back on again. Now is a good time to type in the Basic portion of the program, as given in the article. Refer back to Part A (after the POKE line) for directions on this, and come back here after you've SAVEd and VERIFYed "C64 AID.BAS".

You should now reset the machine again to start with the memory wiped clean. Now we need to make some room to store our code where Basic would normally want to find it. Type:

POKE 8192, 0 : POKE 44, 32 : NEW <RETURN>

Thanks to Mr. Butterfield we have a "one line editor" which is really a poor man's machine language monitor. His TINY PEEKER/POKER was published to ease the burden for people trying to get SUPERMON64 into their C64. Even if that project was not attempted, the program can come in handy for this task. Either drag out your copy, or consult the enclosed listing. My changes were as follows: the variable "T" was changed to "T1" in lines 300, 330 and 430; line 105 was added; and line 210 was dropped. These changes were made so that it will provide a running total of the checksum which may help to confirm your work as you go along.

ReSAVE the PEEKER program if you wish, but we need this in memory for the next step.

RUN the program, and in answer to the INPUT prompt, you should enter "0980" which is the first line in the enclosed tables of hexadecimal values. Continue the entry by typing the other values on the line (using the same spacing shown), then hit <RETURN> to get the next prompt. You then type the "address" and "data" from the second line of the first table. After the sixteen lines have been done, copy down the number shown as the "checksum". This will come in handy later on.

I would recommend that you then exit the program (type "*" after the prompt) and then RUN it again for each of the eleven data blocks. This will reset the checksum counter to zero each time so you can compare the values that appear later in the article.

After all the data is in place, we need to do the following very carefully, so pay close attention and DO NOT turn off your machine yet:

The checking of the ML code is a little more involved. In an ideal case, you got everything right using the PEEKER and your eleven checksums agree perfectly with the DATA values given in the Check–Out section below. This is highly unlikely because the checksum counter would get messed up if you had to re–enter a line in a block of code. Some of the blocks could be OK.

Go to the Check–Out section, which will require you forget

the PEEKER program, by typing 'NEW' before you enter the checksum program. Note the incorrect blocks, because you will have to reLOAD the PEEKER to check and change the erroneous bytes. Then back to the Check-Out until it's finally correct. No one promised that our tiny monitor was going to be the ideal way!

Now, with all OK, let's get the start-of-Basic back where it belongs. Type:

POKE 2048, 0 : POKE 44, 8 : NEW <RETURN>

Now LOAD in the "C64 AID.BAS" program from earlier. In order to include the code we've POKEd in, we have to adjust a few other memory pointers. Do this by typing:

POKE 45, 0 : POKE 46, 15 : CLR <RETURN>

A normal Basic SAVE will unite the two pieces. Call it "C64 AID.REL" and VERIFY at least one copy.

Go to the section on Final Checking.

Check-Out

The following program can be entered to provide some confirmation that you have got the ML portion right. SAVE this one too:

```
10 t=0: for i= 2432 to 3839: t=t+peek(i): next: print t
20 t=0: for i= 0 to 10: s= 2432+i*128
30 t=0: for j= s to s+127: t=t+peek(j): next j
40 read c: if c=t then 60
50 print "block";i+1;"wrong, total =";t
60 next i
70 data 14566, 17673, 14097, 15530, 15344, 14312
80 data 16881, 15633, 13974, 14760, 15154
```

When RUN, the first number shown should be 167924, which is the sum of all bytes in the program. Each block of 128 bytes is then summed individually and compared to the correct value from the DATA statements. Any incorrect block total is identified by number and its total is also printed.

Chances are that at least one correction will be required, so you will have to re-enter the monitor (or reLOAD the PEEKER program) to find the errors. When the detective work is complete, don't forget to reSAVE the machine language code, using the appropriate instructions for the method you've been using.

Now go back to the place where you came from.

Final Checking

All three methods should converge to this spot for the acid test. As usual start with a clear machine before you LOAD "C64 AID.REL". RUN it and the screen should clear, and your Basic program with all the neat documentation (you didn't leave out the credits, did you?) will do its thing.

The cursor should return below the "READY." message. If it doesn't, promise yourself never to undertake such an effort again for one crummy program.

Try out a few of the commands on the program in memory. Don't do APPEND, since we know there's ML code hiding after Basic. If it seems to work fine, then congratulations are in order. Call your best friend and bore him silly with your success story.

For the unfortunate few whose program doesn't work, you can still send me a tape or disk, following the instructions given before. If you wish, send along the non-working copy, and I'll try to find the error(s).

When VIC AID was published, I got a few letters from people who told me that you can use abbreviations, just like with Basic keywords, to minimize the typing. Since C64 AID works the same way, in forcing the machine to examine its commands first, of course we can do the same thing. If you don't know about the "short-form" of Basic keywords, the manual that comes with the VIC or C64 has a table of them. Suffice it to say that we veteran PETters have been lazy since the day we found them.

The Commodore 64 is a very fine machine. I hope you find the added commands useful in your programming efforts.

Tiny-Aid 64: BASIC Portion

```
 1 print "■S■" tab(14) "■r■c64 tiny aid■q■
 2 print "adapted for c64 by: david a. hook
 4 print "■q■from 'tiny aid' by: jim butterfield
 6 print "■q■and 'basic aid' by: bill seiler
 8 print "■q■" tab(12) "■r■ sample commands:
 9 print "■q■change /?/print#4,/
10 print "■q■find .gosub., 200-
11 print "■q■delete 130-625
12 print "■q■number 100,5
13 print "■q■append "chr$(34) "name" chr$(34)", [device
   #]
14 print "■q■kill
15 sys(peek(43)+peek(44)*256+383)
```

```
 70 rem j. butterfield -- compute! jan '83
 80 rem slightly modified by d.a. hook
 90 rem as of april 3/83
100 print "tiny peeker/poker"
105 t=0
110 x$="*" : input x$ : if x$="*" then end
120 gosub 500
130 if e then 280
140 a=v
150 if j>len(x$) then 300
160 for i=0 to 7
170 p=j : gosub 550
180 c(i)=v
190 if e then 280
200 next i
220 for i=0 to 7
230 poke a+i,c(i)
240 t=t+c(i)
250 next i
260 print "checksum=";t
270 goto 110
280 print mid$(x$,1,j);"??" : goto 110
300 t1=0
310 for i=0 to 7
320 v=peek(a+i)
330 t1=t1+v
340 v=v/16
350 print " ";
360 for j=1 to 2
370 v%=v
380 v=(v-v%)*16
390 if v%>9 then v%=v%+7
400 print chr$(v%+48);
410 next j
420 next i
430 print "/";t1
440 goto 110
500 p=1
510 l=4
520 goto 600
550 p=j
560 l=2
600 e=0
610 v=0
620 for j=p to len(x$)
630 x=asc(mid$(x$,j))
640 if x=32 then next j
650 if j>len(x$) then 790
660 p=j
670 for j=p to len(x$)
680 x=asc(mid$(x$,j))
690 if x<>32 then next j
700 if j-p<>l then 790
710 for k=p to j-1
720 x=asc(mid$(x$,k))
730 if x<58 then x=x-48
740 if x>64 then x=x-55
750 if x<0 or x>15 then 790
760 v=v*16+x
770 next k
780 return
790 e=-1
800 return
```

```
.: 0980 a5 2d 85 22 a5 2e 85 23
.: 0988 a5 37 85 24 a5 38 85 25
.: 0990 a0 00 a5 22 d0 02 c6 23
.: 0998 c6 22 b1 22 d0 3c a5 22
.: 09a0 d0 02 c6 23 c6 22 b1 22
.: 09a8 f0 21 85 26 a5 22 d0 02
.: 09b0 c6 23 c6 22 b1 22 18 65
.: 09b8 24 aa a5 26 65 25 48 a5
.: 09c0 37 d0 02 c6 38 c6 37 68
.: 09c8 91 37 8a 48 a5 37 d0 02
.: 09d0 c6 38 c6 37 68 91 37 18
.: 09d8 90 b6 c9 bf d0 ed a5 37
.: 09e0 85 33 a5 38 85 34 6c 37
.: 09e8 00 aa aa bf a9 4c 85 7c
.: 09f0 ad fe ff 00 85 7d ad ff
.: 09f8 ff 00 85 7e 4c 40 fc 00
```

```
.: 0a00 f0 03 4c eb ff 00 a9 c9
.: 0a08 85 7c a9 3a 85 7d a9 b0
.: 0a10 85 7e 60 85 8b 86 97 ba
.: 0a18 bd 01 01 cd fc ff 00 f0
.: 0a20 10 d0 02 a4 8c a6 97 a5
.: 0a28 8b c9 3a b0 03 4c 80 00
.: 0a30 00 60 bd 02 01 cd fd ff
.: 0a38 00 d0 ec a5 8b 10 02 e6
.: 0a40 7a 84 8c a2 00 00 86 a5
.: 0a48 ca e8 a4 7a b9 00 00 02
.: 0a50 38 fd a6 ff 00 f0 13 c9
.: 0a58 80 f0 13 e6 a5 e8 bd a5
.: 0a60 ff 00 10 fa bd a6 ff 00
.: 0a68 d0 e4 f0 be e8 c8 d0 e0
.: 0a70 84 7a a5 a5 0a aa bd c8
.: 0a78 ff 00 48 bd c7 ff 00 48
```

```
.:0a80 20 99 fb 00 4c 73 00 00
.:0a88 20 63 fd 00 a5 5f a6 60
.:0a90 85 24 86 25 20 d9 ff 00
.:0a98 a5 5f a6 60 90 0a a0 01
.:0aa0 b1 5f f0 04 aa 88 b1 5f
.:0aa8 85 7a 86 7b a5 24 38 e5
.:0ab0 7a aa a5 25 e5 7b a8 b0
.:0ab8 1e 8a 18 65 2d 85 2d 98
.:0ac0 65 2e 85 2e a0 00 00 b1
.:0ac8 7a 91 24 c8 d0 f9 e6 7b
.:0ad0 e6 25 a5 2e c5 25 b0 ef
.:0ad8 20 d3 ff 00 a5 22 a6 23
.:0ae0 18 69 02 85 2d 90 01 e8
.:0ae8 86 2e 20 dc ff 00 6c 02
.:0af0 a0 20 d6 ff 00 20 73 00
.:0af8 00 85 8b a2 00 00 86 49
```

```
.:0b00 20 3d fd 00 a5 a5 c9 00
.:0b08 00 d0 07 a2 02 86 49 20
.:0b10 3d fd 00 20 73 00 00 f0
.:0b18 03 20 e8 ff 00 20 63 fd
.:0b20 00 a5 5f a6 60 85 7a 86
.:0b28 7b 20 e5 ff 00 d0 0b c8
.:0b30 98 18 65 7a 85 7a 90 02
.:0b38 e6 7b 20 7d ff 00 f0 05
.:0b40 20 8d fd 00 b0 03 4c 40
.:0b48 fc 00 84 55 e6 55 a4 55
.:0b50 a6 31 a5 32 85 8b b1 7a
.:0b58 f0 d8 dd 00 00 02 d0 ed
.:0b60 e8 c8 c6 8b d0 f1 88 84
.:0b68 0b 84 97 a5 49 f0 5b 20
.:0b70 a1 fd 00 a5 34 38 e5 32
.:0b78 85 a7 f0 28 c8 f0 ca b1
```

```
.:0b80 7a d0 f9 18 98 65 a7 c9
.:0b88 02 90 40 c9 4b b0 3c a5
.:0b90 a7 10 02 c6 8b 18 65 0b
.:0b98 85 97 b0 05 20 d5 fd 00
.:0ba0 f0 03 20 bd fd 00 a5 97
.:0ba8 38 e5 34 a8 c8 a5 34 f0
.:0bb0 0f 85 8c a6 33 bd 00 00
.:0bb8 02 91 7a e8 c8 c6 8c d0
.:0bc0 f5 18 a5 2d 65 a7 85 2d
.:0bc8 a5 2e 65 8b 85 2e a5 7a
.:0bd0 a6 7b 85 5f 86 60 a6 43
.:0bd8 a5 44 20 ee fd 00 20 e1
.:0be0 ff a9 00 00 85 c6 a4 97
.:0be8 4c a3 fc 00 a4 7a c8 94
.:0bf0 31 a9 00 00 95 32 b9 00
.:0bf8 00 02 f0 15 c5 8b f0 05
```

```
.:0c00 f6 32 c8 d0 f2 84 7a 60
.:0c08 c9 ab f0 04 c9 2d d0 01
.:0c10 60 4c eb ff 00 90 05 f0
.:0c18 03 20 57 fd 00 20 e2 ff
.:0c20 00 20 d9 ff 00 20 79 00
.:0c28 00 f0 0b 20 57 fd 00 20
.:0c30 73 00 00 20 e2 ff 00 d0
.:0c38 e0 a5 14 05 15 d0 06 a9
.:0c40 ff 85 14 85 15 60 20 7d
.:0c48 ff 00 85 43 20 7d ff 00
.:0c50 85 44 38 a5 14 e5 43 a5
.:0c58 15 e5 44 60 a5 7a 85 22
.:0c60 a5 7b 85 23 a5 2d 85 24
.:0c68 a5 2e 85 25 60 a5 22 c5
.:0c70 24 d0 04 a5 23 c5 25 60
.:0c78 a4 0b c8 b1 22 a4 97 c8
```

```
.:0c80 91 22 20 b2 fd 00 d0 01
.:0c88 60 e6 22 d0 ec e6 23 d0
.:0c90 e8 a4 0b b1 24 a4 97 91
.:0c98 24 20 b2 fd 00 d0 01 60
.:0ca0 a5 24 d0 02 c6 25 c6 24
.:0ca8 4c d5 fd 00 a0 00 00 84
.:0cb0 a5 84 0f 20 ee ff 00 a9
.:0cb8 20 a4 a5 29 7f 20 d2 ff
.:0cc0 c9 22 d0 06 a5 0f 49 ff
.:0cc8 85 0f c8 b1 5f f0 19 10
.:0cd0 ec c9 ff f0 e8 24 0f 30
.:0cd8 e4 84 a5 20 2d fe 00 c8
.:0ce0 b1 ae 30 d6 20 d2 ff d0
.:0ce8 f6 20 e5 ff 00 38 60 ac
.:0cf0 fa ff 00 84 ae ac fb ff
.:0cf8 00 84 af 38 e9 7f aa a0
```

```
.:0d00 00 00 ca f0 ec e6 ae d0
.:0d08 02 e6 af b1 ae 10 f6 30
.:0d10 f1 20 e2 ff 00 a5 14 85
.:0d18 35 a5 15 85 36 20 e8 ff
.:0d20 00 20 e2 ff 00 a5 14 85
.:0d28 33 a5 15 85 34 20 df ff
.:0d30 00 20 7d ff 00 20 7d ff
.:0d38 00 d0 21 20 5f ff 00 20
.:0d40 7d ff 00 20 7d ff 00 d0
.:0d48 03 4c 40 fc 00 20 7d ff
.:0d50 00 a5 63 91 7a 20 7d ff
.:0d58 00 a5 62 91 7a 20 6a ff
.:0d60 00 f0 e2 20 7d ff 00 20
.:0d68 7d ff 00 20 7d ff 00 c9
.:0d70 22 d0 0b 20 7d ff 00 f0
.:0d78 c5 c9 22 d0 f7 f0 ee aa
```

```
.:0d80 f0 bc 10 e9 a2 04 dd a1
.:0d88 ff 00 f0 05 ca d0 f8 f0
.:0d90 dd a5 7a 85 3b a5 7b 85
.:0d98 3c 20 73 00 00 b0 d3 20
.:0da0 e2 ff 00 20 04 ff 00 a5
.:0da8 3c 85 7b a5 3b 85 7a a0
.:0db0 00 00 a2 00 00 bd 00 00
.:0db8 01 c9 30 90 11 48 20 73
.:0dc0 00 00 90 03 20 35 ff 00
.:0dc8 68 a0 00 00 91 7a e8 d0
.:0dd0 e8 20 73 00 00 b0 08 20
.:0dd8 44 ff 00 20 79 00 00 90
.:0de0 f8 c9 2c f0 b8 d0 96 20
.:0de8 5f ff 00 20 7d ff 00 20
.:0df0 7d ff 00 d0 08 a9 ff 85
.:0df8 63 85 62 30 0e 20 7d ff
```

```
.:0e00 00 c5 14 d0 0f 20 7d ff
.:0e08 00 c5 15 d0 0b 20 f1 ff
.:0e10 00 a9 20 4c d2 ff 20 7d
.:0e18 ff 00 20 6a ff 00 f0 d2
.:0e20 20 55 ff 00 e6 97 20 d5
.:0e28 fd 00 e6 2d d0 02 e6 2e
.:0e30 60 20 55 ff 00 c6 97 20
.:0e38 bd fd 00 a5 2d d0 02 c6
.:0e40 2e c6 2d 60 20 a1 fd 00
.:0e48 a0 00 00 84 0b 84 97 60
.:0e50 a5 35 85 63 a5 36 85 62
.:0e58 4c df ff 00 a5 63 18 65
.:0e60 33 85 63 a5 62 65 34 85
.:0e68 62 20 7d ff 00 d0 fb 60
.:0e70 a0 00 00 e6 7a d0 02 e6
.:0e78 7b b1 7a 60 a9 00 00 85
```

```
.:0e80 0a 20 f4 ff 00 38 a5 2d
.:0e88 e9 02 aa a5 2e e9 00 00
.:0e90 a8 a5 0a 20 d5 ff 4c f7
.:0e98 ff 00 89 8a 8d a7 43 48
.:0ea0 41 4e 47 c5 44 45 4c 45
.:0ea8 54 c5 46 49 4e c4 4b 49
.:0eb0 4c cc 4e 55 4d 42 45 d2
.:0eb8 41 50 50 45 4e c4 00 00
.:0ec0 56 fc 00 f2 fb 00 56 fc
.:0ec8 00 77 fb 00 4b fe 00 87
.:0ed0 ff 00 4c 33 a5 4c 7c a5
.:0ed8 4c 13 a6 4c 59 a6 4c 8e
.:0ee0 a6 4c 6b a9 4c d7 aa 4c
.:0ee8 fd ae 4c 08 af 4c cd bd
.:0ef0 4c d1 bd 4c d4 e1 4c 95
.:0ef8 e1 9d a0 8c a4 8a fb 00
```

1982 Commodore Bibliography

Don White
Ottawa, Ont.

Looking for an article but don't know where to start? Don White of the Ottawa Commodore Users Group has compiled the following bibliography of Commodore related articles published in 1982 issues of Creative Computing, Kilobaud MICROCOMPUTING and COMPUTE!. These are three of the more popular mags on micros and although they may not have back copies for sale, your local computer club might be able to help you find a particular issue if you're in a bind. Ed.

Creative Computing

January

42 Five for the VIC-20 – Brief reviews of Car Chase, Slither, Super Slither, Casino-Style Blackjack, Blue Meanies From Outer Space, and Biorhythm Compatibility

126 Big Numbers and Small Computers – Program to simulate a calculator that can add, subtract, multiply, divide and raise to a power integers of up to 1024 decimal digits.

February

36 New Graphics Horizons For The PET – Review of MTU's Visible Memory Board.

58 More On VIC Graphics – Brief description of VIC graphics capabilities.

148 Graphics Conversion For The TRS-80, Apple, and PET – Information regarding the graphics capabilities of the 3 computers and ways of converting graphics routines.

200 Personal Electronic Transactions – Educational Programs by Don Ross; Programs by Teaching Tools; BSR Wars Continued

April

186 Data Without Duplicates – Routine to allow random selection of data statements.

May

118 Autohex PEEK/POKE – A program to convert machine language programs into Basic DATA statements.

June

182 Maze Race – Two-player car race for the PET.

222 Personal Electronic Transactions – 2020 vs 4040; Basic 3.0 vs. Basic 4.0; IEEE-488 Extension Card; XDOS by Prominico; The +20 ROM; Hex to Decimal Infinitum; The Nuclear PET.

July

162 Screensaver – Program that permits drawing or printing on the screen, saving the screen in DATA statements and finally, returning the original screen.

208 Personal Electronic Transactions – Cheapskate disk filer; Uncopyable Tapes; A reasonable GET routine; the Pedisk II; The User Port Workshop; an Eclectic ROM – Faster Basic; The PIC Chip.

August

171 Droids : A strategy Game for the PET – An educational game for up to 4 players.

236 Personal Electronic Transactions – Some VIC Modifications; JINSAM; Fixing Flaky Diskettes; An 8050 Disk Bug; VIC and the Frying Pan; Entry Routine for Musical Chords.

September

212 Personal Electronic Transactions – Two little programs for a printer; Benchmark Programs; The Petspeed Compiler; A Quick and Dirty Program; PET/CBM Basic.

October

14 Math For Older Students – Review of Factoring Whole Numbers.

298 Personal Electronic Transactions – PetChess; PET Nuke Fix; CP/M for the PET; VIC Bits; The Single Disk Glitch; DiscSavers; A Write Protect Trick; Asteroidz and Munchman; Pakjana; Filemaster; A Yeech Program.

November

294 Personal Electronic Transactions – Programming the PET/CBM; VIC Revealed; Nine Ways to Write a Program Languages for the PET; PET/CBM Basic; Comal; Petspeed Compiler; Oxford Integer Basic Compiler; KMMM Pascal; Forth; RPL; Vigil; 6502 Assembler; About Adventure and FRP Games; Dungeon of Death; Pagoda; Explore; Dunjonquest; Micro Warrior; SwordQuest; Death Star.

December

326 A Scrolling Routine for CBM/PET Output Listings – Routine allows scrolling of output from a program.

396 Personal Electronic Transactions – VIC Product Guide

Kilobaud MICROCOMPUTING

January

9 PET–Pourri – 80–Column Adapter; Formatted disk files from WordPro; VIC jump vectors; Disk–To–Tape Data File Copy.

102 A Computer/Video Disk Combo That Really Works – Interfacing a PET to a Pioneer video disk player.

132 The Revealing Truth About PET's Memory – A BASIC text disassembler.

146 Putting PET To The Test – Adaptation of Cook's memory test for the 6502.

206 Flash Attack – A review of a multimachine war game.

February

10 PET–Pourri – Reverse Polish Language (RPL).

March

12 PET–Pourri – DTACK Grounded; Multipurpose Interface; EHS's Scroll; VIC Budget.

April

12 PET–Pourri – VIC–1540 Disk; Input Bug; Basic Aid; Micromon; ATUG; WordPro Quit.

May

12 PET–Pourri – UCSD Pascal; VIC Programmers Reference Guide; Teach Yourself Programming Series; Home Calculation Six–Pack; VIC Cartridge Games; VICModem; VIC Memory Expansion; New Machines; Winchester Disks; VIC RS–232 Notes; WordPro Word Counter.

June

12 PET–Pourri – Programming The VIC–20.

56 Disk Master – Disk cataloguing program.

July

14 PET–Pourri – WordPro Splitter; Wordcraft Utility; New VIC–20 Software; Misc.

August

13 PET–Pourri – Hescount; Hescat; Hescom; Commodore News; Educational Software; Misc.

44 Fowl Play – Using PET computers to study animal behaviour.

September

8 PET–Pourri – STCP; Hex Dump; Misc.

88 Black Friday – A stock market simulation for one to four players.

100 Dueling Joysticks – Add two more joysticks to your VIC–20.

104 The Game Room : Invaders, Pac–Man Games Predominate – A review of various arcade games including Snakman for the VIC–20

October

20 PET–Pourri – Commodore News; On Line Software; Misc.

32 What You Didn't Know About NEC Spinwriter – Using a PET/CBM and RPL to plot on a NEC Spinwriter.

88 Squeeze The Most Out Of Your VIC–20 – Use the colour dot and high–resolution capabilities to generate super game graphics.

152 Conversions "II" – "Color Code Combo" converted for the PET.

November

19 PET–Pourri – Time Is Money; PET Graphics; Misc.

42 Micros Find A Place Under The Sun – CBM 8032 used to collect and analyse data on the efficiency and usage of hot water solar systems.

December

26 PET–Pourri – Disk Tips; Disk Master Additions; HES Software For The VIC; Newsletters

30 Micro Software Digest – Solicube.

74 A Big Boost For First–Time Users – A review of the VIC–20.

118 Conversions – A conversion of "Micro Money Maker" for the PET/CBM.

173 The ARROW – A review a utility package in EPROM that provides high speed save, load, verify and append with cassette, tape positioning, repeat–key function, character set flip, hexadecimal mode, and quadruple density plotting.

COMPUTE!

January

14 Ask The Readers – Note On Chaining Programs.

54 Anti–Hesitation Programming : A Tutorial On Arrays – A discussion of programmimg methods to speed up execution of programs.

144 Renumbering An Appended Routine Only – Using Toolkit to renumber an appended routine.

146 BRANCH NEVER And QUIF Assembling On SuperPET – A discussion of the 6809 Assembler on the SuperPET.

150 PET Repairs For The Amateur – Loose connections and dirty contacts are the most frequent source of problems on early PETs.

152 Realtime Clock On Your PET Screen – A machine language routine to continuously display the time on the PET screen. (Upgrade & 4.0)

156 Tape Load Test And Head Alignment – How to prepare and a use a special test tape for the cassette recorder of any PET/CBM.

160 MicroMon – An Enhanced Language Monitor – Machine language monitor including a simple assembler, disassembler, compare, hunt, transfer, relocater, single step and various other options.

174 Self–Modifying Programs In BASIC – How to write self–modifying programs.

176 TinyMon : A Simple Monitor For The VIC – A monitor which displays the registers, displays memory, saves and loads programs and executes a program.
180 VIC Color Tips – A discussion of using color on the VIC.
181 VIC Memory Map Above Page Zero – More Butterfield memory maps.
183 ZAP!! – A VIC game for up to 6 players with 5 rounds per player.
186 CAPUTE! – Compute! #17, pg. 143; Compute! #17, pg. 152.

February

28 Insurance Inventory – A program to maintain an inventory of personal possessions on cassette
36 Creating A Simple Word Processor – A word processor program in BASIC for the PET/CBM.
44 Transposition – Some simple routines to demonstrate the transposition of music on the PET. Uses CB2 sound.
52 Some Common Basic Programs – A book review.
54 Multitask : A Realtime Multitasking Operating System Emulator – A Basic program to demonstrate multitasking on a PET.
128 A User–Defined Character Editor – A program for creating a user–defined character set for use with a printer having the capability of printing user–defined characters.
135 Marquee – A machine language routine that will continually scroll text hoizontally on the PET/CBM screen.
145 Disk Disassembler – This program disassembles code from a file on disk and dumps the output to a printer.
154 Line Input For The PET – A machine language routine to add a LINPUT command to the PET.
160 Measure Time Intervals With The PET Parallel Port – A machine language routine that can be used to measure 7 successive small time intervals to the nearest 1/10000 s.
166 Screen Pro – A review of a screen utility package.
168 Extended VIC–20 Input Devices : Paddles And The Keyboard – A discussion of the VIC–20 paddles and keyboard with examples.
173 Timekeeping – Using the TI and TI$ variables on the VIC.
176 An Easy Way To Relocate VIC Programs On Other Commodore Computers – Two methids of easily relocating VIC programs.
177 UMI Amok For VIC – A game review.
178 UMI 3K VIC Memory Exapansion – A hardware review.
179 Alphabetizer – A simple sorting routine.
181 CAPUTE! – The Unwedge; Bits, Bytes And Basic Boole; Assembler Update; Inversion Partitioning.
183 New Products – Board Offers 24K Additional Memory For VIC; TYCOM Introduces 3 Additional Educational Packages(Algebra Word Problems, Spanish, German); Multi–Purpose Interface For PET/CBM; From Krell Software Corp.(War Of The Samurai, Alexander The Great, Isaac Newton, Fig Newton, Odyssey In Time).

March

12 Ask The Readers – Odds And Ends; VIC–20 Tips.
62 Infinite Precision Multiply – An infinite precision multiplying routine in BASIC.
68 Word Hunt – A game to find specific word or letter sequences in a 10 by 10 matrix.
78 Count The Hearts – A program to help develop a child's counting ability. (VIC)
88 Family : A Simulation In Genetics – A program to demonstrate the effects of gene selection.
104 Large Alphabet For The VIC – A short routine to display double–size alphabet on the VIC.
112 Starfight3 – Startrek for the VIC.
117 Webs – A one–player game for a 40–column PET.
120 RPL : A FORTH Sequel – A brief description of Reverse Polish Language from Samurai Software.
139 Part I : Disk Checkout For 2040, 4040, And 8050 Disks – An explanation of disk manipulations via machine language.
152 Dynamic Renumber – A Basic utility to renumber a selected range of lines.
168 More VIC Maps – Butterfield's memory maps for the VIC.
182 Random Music Composition On The PET – A program to let the PET compose and play music.
194 Basic 4.0 To Upgrade Conversion Kit – A discussion on converting Basic 4.0 programs to Upgrade ROM.
209 New Products – VIC–20 Timesharing With Printout; Scratchpad Mailing List; VicModem.
222 CAPUTE! – Spacewar Part 2; PET To PET Communications Over The User Port; Tinymon1.

April

12 Ask The Readers – Simple Word Processor; Tape Index Program; Self–Modifying Programs.
26 Moving Averages – A program to calculate moving averages.
34 Track Down Those Memory Bugs – A sophisticated machine language memory testing program.
50 Shooting Stars – A short game for the PET/CBM.
70 Grading Exams On A Microcomputer – A program to assist in grading exams.
84 Using The VIC Game Paddles – A tutorial on how the game paddles work on a VIC–20 illustrated by the game, Breakout.
98 Micros With The Handicapped – Communicating with a micro.
100 Using The PET/CBM In The High School Physics Lab – Programs which allow you to measure frequencies or time intervals in microseconds using phototransistors and a PET/CBM.
108 Intelligent Input Routines – Routines that try to correct faulty user input.
122 Machine Language : Jump To It – On using Jump Tables.
126 PETASCII To ASCII Conversion – A machine language

routine to convert PETASCII to ASCII. (PET/CBM)

135 Odds And Ends – The Bug In The Universal Wedge; Improving The Toolkit's TRACE Function.

142 Browsing The VIC Chip – An explanation of the 6560/6561 registers with experiments. (VIC)

155 Extending MAE – How to add custom pseudo–ops.

156 Part II : Disk Checkout For 2040, 4040, And 8050 Disks – A program to perform quality checks on disks and recover scratched disks.

174 A High Resolution Digital–To–Analog Converter For The PET – The necessary hardware and software to do D–to–A conversions on the PET.

179 The "Branding Iron" EPROM Programmer For The PET/CBM – A review.

181 VIXEL #1 – A review of the first VIC three–in–one package from The Code Works.

182 CAPUTE! – VIC : Alternate Screens; MICROMON; Timekeeping.

183 New Products – Educational Aids For The Classroom; Understanding Your VIC From TIS; Use Graphics And Sound For Blackjack Program; Multi–purpose Interface For CBMs; Communication And Cataloging Now For CBM; PET Terminal Emulator; Symtec Light Pen For The Atari And VIC; Software For Elementary Students; Programs For The Classroom; Disks And New Educational Programs From Teacher's Pet Software.

May

12 Ask The Readers – A User–Defined Character Editor; Commodore 4040 Drives; VIC–20 Tape Drives; Bullet-proof INPUT; VIC–20 Manual Correction.

32 Life Insurance Estimator – Analyse your life insurance needs. (PET/VIC)

42 Some Speculation On The Well–Programmed Game – An approach to good programming and development of a simple game.

52 A New Technique For Mixing Basic And Machine Language – On reserving a 249–byte or 2739–byte block of memory in your Basic program for machine language. (PET/VIC)

63 Modem Driver Module [MDM–1] – A review of the MDM–1 interface to connect a VIC–20 to a modem and a short Basic driver program.

78 The Joystick Connection : Meteor Maze – An action game program illustrating how to use joysticks with the VIC–20.

98 Amortize – An amortization program with an INPUT technique allowing the input of expressions as well as numbers. (VIC)

110 Machine Language : First Steps : Part I – Developing a machine language program to draw bar graphs on the screen. (PET/CBM)

135 Recovering From NEW On Apple And CBM – How to reset some pointers and recover from NEW. (PET/CBM)

136 Putting The Squeeze On Your VIC–20 : Getting The Most Out Of 5000 Bytes – How to reduce the size of your programs.

156 Screen Input On The PET – A series of experiments illustrating how to avoid "garbage collection" delays when working with pre–Basic 4.0 PETs and CBMs.

160 Fast Sort For PET/CBM – A machine language sorting utility that can be used with tape or disk. (PET/CBM – Upgrade & 4.0)

170 Handicapped Programming – A program that prints messages on the screen and allows the user to move a pointer to the desired word and enter it into the computer. (PET with graphics keyboard)

175 New Products – VIC–20 Programmers Reference Guide; WordPro 5+ and WordPro 2+; Hayden Announces Computer Literacy Package; VidCom Anounces Product To Interact With VTR.

188 CAPUTE! – Renumbering An Appended Routine Only; Starfight3.

June

10 Ask The Readers – Keyprint; VIC–TTY Interface

20 Income Property Report – Prepare information from income property for yearly taxes.

30 Outpost – A stategy game for any Commodore computer.

43 Search For PET And Apple II Plus – A routine to provide a Search utility. (PET/CBM – Upgrade & 4.0)

46 A VIC Intelligent Video Disk System – Interface and software to control the Pioneer VP 1000 Laser Disk with a VIC–20.

52 Some Similarities Between Applesoft and PET Basic – A commentary on the ease of converting PET and Apple programs with some examples.

79 Using Atari Joysticks With Your VIC – An explantion on using the joystick with the VIC and a sample drawing program, Doodle.

87 PET Miscellany – Jim Butterfield responds to a number of queries about Commodore computers.

88 Bits And Pieces – Elizabeth Deal provides informations concerning Commodore machines.

89 PET Newsletters And Magazines – A list of worthwhile publications.

94 Micros With The Handicapped : Developing A Communications Program : Part II – A menu–communicator for Apple, PET and VIC computers.

104 VIC–20 Cartridge Games – A review of Jupiter Lander, Super Alien, VIC Avenger, Draw Poker and Super Slot.

107 Hardbox For PET/CBM – Review.

108 Beyond Games : Systems Software For Your 6502 Personal Computer – Book review.

119 Hooking Up DOS Wedge To PETs With POWER – A procedure that allows the use of DOS support and/or Universal Wedge on Upgrade PETs equipped with the POWER chip.

128 VIC/PET Basic Program Transfers – A few methods for loading Basic VIC programs into a PET.

133 Machine Language : First Steps, Part II – Converting a bar graph program from Basic to machine language.
139 Run 96K Programs On The SuperPET – How to use all 96K on the SuperPET from Commodore Basic.
144 A Simple 2716 EPROM Programmer For The PET – Hardware and software required to burn 2716's with a PET.
151 VIC's Perpetual Calendar – A perpetual calendar program for the VIC.
158 Basic Program Merges : PET And VIC – Routines to merge programs on PETs, CBMs and VICs.
175 New Products – SoftBox, HardBox and Petspeed Compiler; Graphics Package For The VIC–20.
190 CAPUTE! – Micromon

July

6 The Editor's Notes – A Brief Overview Of The Chicago Consumer Electronics Show And National Computer Conference In Houston.
12 Ask The Readers – Basic To Basic; VIC Fast–Find; Hidden Adventure; VIC Expansion Software.
22 The Beginner's Page : Making Files Work – Some specific details about file handling.
27 Gold Rush! – An exciting action game requiring a joystick. (VIC)
34 IRA Planner – Compute your IRA account(U.S.). (PET/CBM/VIC)
40 Maze Race – A one or two player maze game.
58 Recursive Basic Subroutines – A simple game illustrating the concept of recursive subroutines.
68 Screen Graphics – How to move and rotate objects on the screen.
84 Answer Selection With Joysticks – A program to illustrate the use of the joystick to input responses. (VIC)
94 Computing Techniques For The Handicapped – A quadriplegic describes techniques he has learned to effectively work with his PET.
96 Multidigit Addition – A program to allow you to add very large numbers.
97 A Direct Access File Editor – A disk file editing program. (PET/CBM – Update & 4.0)
112 Whither VIC? – Butterfield discusses VIC software, add–ons, and VIC's place in the world of computing.
116 Super QuadraPET – A Basic 4.0 update of QuadraPET, a program to partition the memory of a 32K PET into four 8K areas. (PET/CBM – 4.0)
130 All About PET/CBM Character Sets.
144 VIC Super Expander Memory Map – A map of the significant routines in the VIC Super Expander.
150 Part III : Machine Language : First Steps
154 How To Use The 6560 Video Interface Chip – Technical details on the 6560 VIC.
159 Machine Language Compactor – A routine to compact Basic programs. (PET/CBM – Upgrade & 4.0 – disk)
184 Two Programs From The VIC 6 Pack – Reviews of Car Chase and Blue Meanies From Outer Space.
185 SoftBox–CP/M For PET/CBM – Review.
186 SYSRES For PET/CBM – Review of the SYSRES programmer's aid package.
190 CAPUTE! – Further Notes on Fast Sort For PET/CBM.
191 New Products – Disk–O–Mate For CBM Disk Drives; Budget II From RAK Electronics; Plug In BCD For PET Computers; New Educational Microcomputer Literature From Commodore; Promqueen Cartidge For VIC–20; The VIC Enters The World Of Radiocommunications.

August

10 Ask The Readers – VIC Upgrades; Recover From NEW on VIC; SuperPET Users Group; VIC SuperExpander Hints; VIC Zenith Jitters.
18 The New Wave Of Home Computers – Highlights on the CES and NCC shows.
39 Household Budget Manager – A personal budget program. (PET/VIC)
54 Word Games – A program that creates several different word games on a printer.
60 A First Look At The Commodore 64 – Part 1 of a review of the features and capabilities of the 64.
84 Guess That Animal! Simulating Learning In Computers – A demonstration of how a computer can "learn".
93 Two VIC Word Processing Programs – A review of Un–Word Processor and A VIC Typewriter.
99 VIC Communications : The RS–232 Interface – On interfacing RS–232 modems to your VIC.
103 The Keyprint Compendium – 14 versions of Keyprint.
107 Screen Saver – A Basic routine to transfer screen images to disk and back to the screen.
139 PET Auto Repeat – A repeat–key routine. (PET – 1.0 & Upgrade)
140 VIC Curiosities – Cold Start by SYS 64802; One–handed RUN; LIST killer; SAVE killer.
141 A Light Pen For Under $10 – How to construct a light pen for the VIC–20.
153 Electric Eraser – A routine to delete instructions from a program after they are no longer needed. (VIC/PET/CBM – Upgrade & 4.0)
158 Inner Basic – An explanation of what Basic looks like to your PET and VIC and how to send text to a printer from machine language.
160 Copy 2031 Files – A file copying program for the Commodore 2031 disk drive. (PET/CBM – Upgrade & 4.0)
164 VIC–Key – A routine which allows the user to assign Key–words to each shifted alphabetic key.
175 New Products – Commodore 8300P Printer; File II for PET and VIC.

September

10 Ask The Readers – Lowercase on Comprint; PET to Epson; A New VIC Champion; VIC User Groups.
28 Peripheral Vision Exerciser – Aid in improving and

maintaining speed reading skills.

30 User-defined Functions : Defined – An in-depth explanation of the DEF FN command.

45 Meet Jim Butterfield – He discusses the future of computing, his background and hobbies, and his views on topics ranging from computers in education to the appeal of programming.

56 Banish INPUT Statements! – Some suggestions and substitutions for Basic's INPUT statement.

74 Student Mark Adjustment – A set of programs to calculate accurate adjustments.

78 The Inside World Of The Computer : The Talking Head – A program to create a "talking head" on the computer screen.

92 The Statistics Page : Accurate Statistical Calculations – A method for achieving statistical accuracy on microcomputer systems

101 FlexFile : A PET Data Base Manager – Review

106 Sprite Graphics And Sound Synthesis On The Commodore 64 – Part 2 of an overview of the 64 with game programming examples.

111 PET Pointer Sort – A fast, effective sort that operates on DATA statements or arrays. (PET/CBM – Upgrade & 4.0)

127 PET Autoload – A routine to automatically LOAD and RUN any program selected from the directory. (CBM)

132 $20 VIC Digitizer – Construct a digitizer that works through the VIC game port.

136 On-The-Spot Commodore Disk Fixes – Emergency procedure when disk problems occur.

139 VIC Sticks – A routine for using joysticks with the VIC.

159 VIC Pause – A routine to allow pausing of a program listing.

164 A VIC Bug – Suggestions for overcoming the problem with the VIC INPUT statement.

166 Three PET Innovations – Routines to display both character sets simultaneously, hide PRINT statements and generate a flashing cursor with the GET statement.

174 PET Machine Language Delete – A routine to allow deleting of ranges of lines in PET programs. (PET/CBM – Upgrade & 4.0)

196 CAPUTE! – PET Compactor.

201 New Products – Portmaker From Microtech; Commodore Bulletin Board; Two VIC Games From Computer-Mart (Alien Invasion, Snakeout); Professional Software For Commodore Machines (WordPro-ML, WordPro--Mail List, InfoPro, The Administrator); Disk Based Data Manager for VIC-20; Plotting Routine For VIC.

October

10 Ask The Readers – Butterfield On RS-232 Interfacing.

32 Writing Your First Game – Programming the old card game 'High Card'.

36 Programming Games On Computers With Limited Memory – Suggestions for programming on the VIC and other small computers.

44 Meteor Storm – Navigate your starship through oncoming meteors. (PET/CBM)

50 Rubik's Cube Solved – Program to solve the infamous cube step-by-step. (PET/CBM)

66 Superchase – Try to eat all the treasures before the monster eats you. (5K VIC)

72 Mathman – Game that teaches math. (VIC)

76 TAG – Play tag against the computer. (PET/CBM – Upgrade & 4.0)

86 Laser Barrage – Defend your energy pods against the 15 amok robots. (PET/CBM)

124 Micros With The Handicapped – VIC game demonstrating how the motor-impaired can communicate using only the button on a joystick.

132 Four New Cartridges For VIC-20 – Review of Omega Race, Gorf, Sargon II Chess and Visible Solar System.

136 CURSOR : Issues 23 Through 28 – Classification of CURSOR programs for educational purposes.

139 PET Fun And Games – Book review.

141 Pixelator – Programs to design custom characters for VIC.

150 Commodore 64 Memory Map

156 The VIC Keyboard Redefined – With this short program you can make any key on the keyboard represent any other key.

162 Pack Up Your DATA – Packing data in order to save space in sequential files. (PET/CBM/VIC)

164 PET Tape Head Alignment – Method of aligning tape heads by listening to the signal.

170 PET Self-Starting Programs – Bootfixer routine to make any program loaded from disk self-starting.

173 VIC Joystick And Keyboard Routine – Gaming routines to help speed up Basic.

193 Digital Speech – Technique to store and play digitized speech. (PET/CBM – Upgrade & 4.0)

198 VIC Ringer – Adding an end-of-line bell to the VIC.

201 Is Anyone Open? – Utility to avoid the problem of improperly CLOSEd PET/CBM files.

204 Sorting By Fields – This ripple sort will sort records using any internal locations as its key. (PET/CBM/VIC)

211 CAPUTE! – Machine Language: First Steps; VIC Curiosities.

215 New Products – Expansion Chassis For VIC; Publications From K-12 MicroMedia; Software For Gifted And Talented Students; Contest Marks Merger; VIAC:The VIC Interface To Any Cassette; Grades Management System For Teachers; Commodore Announces Bilingual Keyboard For Microcomputers.

November

14 Ask The Readers – Color For VIC Gold Miner; A VIC Taping Mystery Solved; VIC Superexpander Graphics; SuperPET Users Groups; Running 40-Column Programs On The PET/CBM.

44 Laser Gunner : Basic Animation – Arcade style game-.(PET – Upgrade & 4.0)

56 UXB – Defuse bombs without tripping a mine. (VIC)
74 VIC Harmony – Teach your VIC to sing in 3–part harmony.
78 Rainbow Clock – Make VIC screen a digital clock.
80 Statistician – Learn and examine statistics.
90 Part 1 : How To Use SYS And USR – Introduction to SYS and passing information between Basic and machine language.
104 VisiCalc Home And Office Companion – Book review.
112 VIC–20 Cartridge Games[VIC Firmware] – Reviews of Spiders Of Mars, Satellites and Meteroites.
115 Petspeed, An Optimizing Compiler For PET/CBM – Review.
146 A Terminal Operating System For PET To HP3000+ – Descibes a terminal operating system from a PET to an HP3000 via a SADI interface and modem.
154 From VIC–20 To Mainframe – A program which can handle a variety of communications needs for the VIC.
160 A Shape Generator For The Commodore 64 – Program to draw a shape, examine it, modify it and save it in DATA statements.
168 Easy File Input : The String Thing – Routine to allow input from disk or tape of strings up to 255 characters in length. (PET/CBM/VIC/64)
172 VIC Micromon – Monitor with 34 commands.
192 Capute! – PET Machine Language Compactor Fix.
194 Machine Language : Serial Comunications – How to telecommunicate with the help of machine language.
196 Programming VIC's Function Keys – Machine language routine to assign keyphrases to VIC function keys.
202 PET : Picture Files – Save/load screen images to/from disk. (PET – Upgrade/CBM 8032)
206 Calling Routine For Marquee – Routine to allow interfacing to Marquee (Feb. 1982) from Basic. (PET/CBM – Upgrade & 4.0)
207 PET Interfacing – Introduction to PET interfacing with a simple example.
223 VIC Plotting – Medium resolution plotting on the VIC.
235 New Products – Interfaces For Commodore 64 And VIC; Job Costing And General Accounting Program; VIC Software from Western New England; A 40/80 Character Expansion For The VIC–20; Terminal Emulation Package For The SuperPET; Fractions Programs For PET, TRS–80, And Apple.

December

10 Ask The Readers – VIC Soft Memory Recovery; Machine Language Printing; Commodore 64 Peripherals.
62 Simulator : Tiny Plan, A Modeling Planner For Home Applications – A computerized spreadsheet program in Basic which will work in 8K without a disk or printer.
84 CalCalc : Computerize Your Diet – A program to help you lose weight by cutting calories.
96 All Sorts Of Basic Sorts – A selection of four different sorting routines in Basic.
114 Part II : How To Use SYS – The conclusion of this tutorial demonstrates how to handle complex multiplications in machine language.
126 Name Play – A user–friendly program aimed at preschoolers. Requies a printer. (PET/CBM/VIC)
138 VIC And PET PILOT Interpreter – A low–budget approach to programming in PILOT. Supports sting($) and numeric(#) variables, and the statements T, J, U, E, M, I, C, A, H and End.
152 Hidden Maze – You are trapped inside a maze and you can see only a short distance along its dark corridors as you try to find your way out.
164 Understanding VIC High Resolution Graphics – This article explores high resolution graphics on the 5K and extended 8K VIC–20.
172 Supergraphics For PET – A review of John Fluharty's Supergraphics software.
179 The VIC "Cardboard" – A review of the Cardboard Expansion Interface.
180 Mikro Chip Assembler For The PET – A review of the Mikro assembler from Skyles Electric Works.
192 A Universal Program Lister – This program will list any program in a way that can be easily understood; all special characters for Commodore computers are taken into account. Requires a disk drive.
208 VIC Block SAVE And LOAD – A program to connect you to the kernal routines and allow the saving or loading of blocks of memory from/to tape.
212 Commodore 64 Sprite Editor – Create and modify multicolored sprites on the Commodore 64.
215 Tiny Aid For VIC–20 – Add renumber, delete, find, change and kill to the VIC.
224 Paper Monitor Switch For 2022 Printer – Add an automatic out–of–paper switch to the Commodore 2022 and 2024 printers.
226 A Floppy With A Strange Device – This routine will change the device number of the 2040, 4040, 8050 or 2031 disk drives.
227 VIC File Clerk – A program that allows you to store and quickly locate up to 60 pages of information on one cassette tape.
236 Codemover – A PET program that allows you to easily move machine language programs from one area of memory to another.
247 Checkbook – A checkbook balancing program.
255 Flashing Prompt For VIC and PET – A handy input routine to make your programs crash–proof.
259 New Products – Light Pen For The VIC–20; SuperPET Upgrade Board For CBM 8032; Printer Programming Manual For VIC And Epson MX–80; Action Games For The VIC–20 (Videomania, Terraguard); PET Joystick Interface; Carrying Case For Commodore 64 And VIC; Game From Avalon Hill (Andromeda Conquest); Communications Package For VersaModem; Spread Sheet For VIC.
284 CAPUTE!–PET Laser Gunner; PET Picture Files; C64 memory map; Micros with the Handicapped(Oct82)

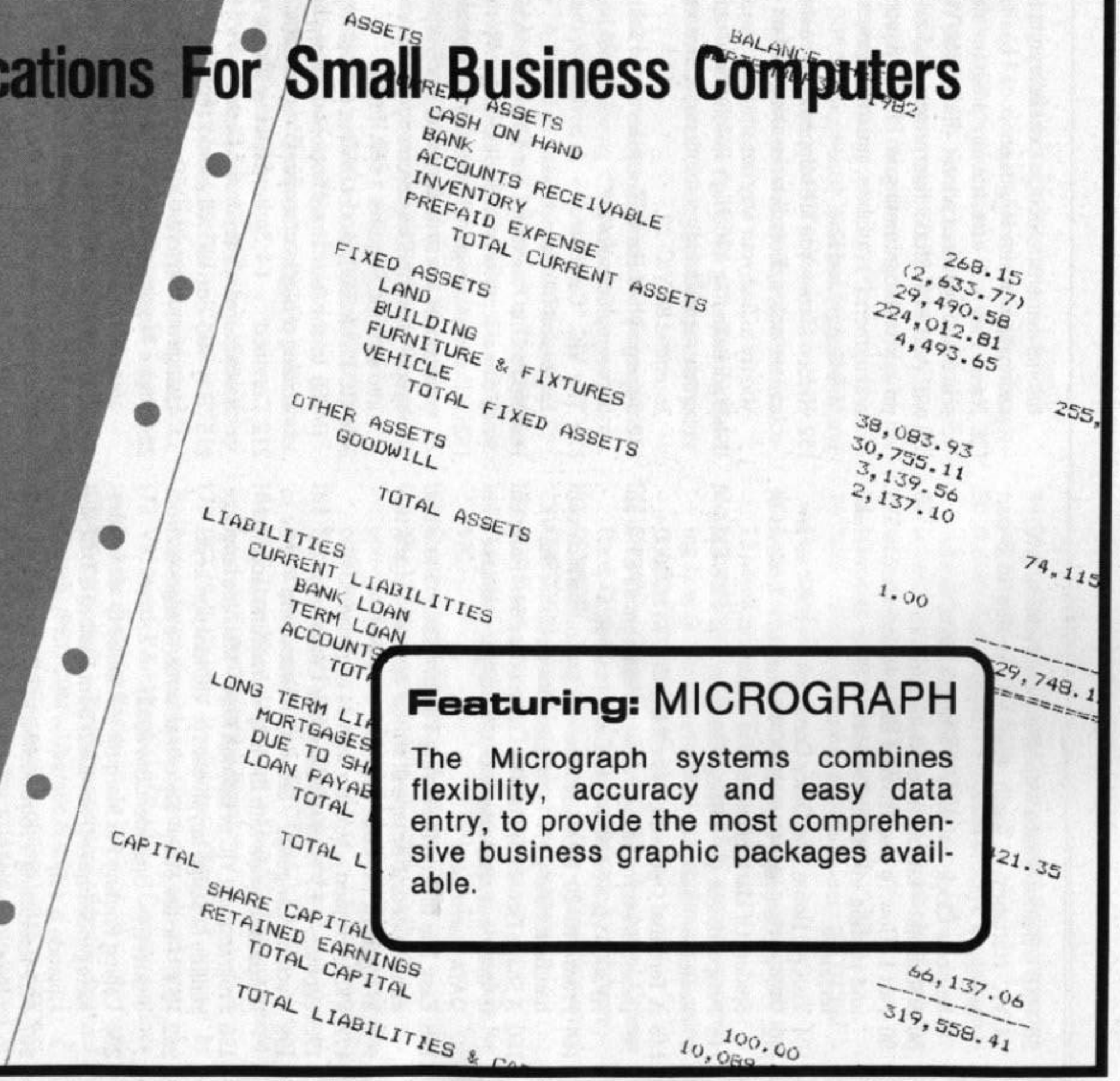
S O F T W A R E

Commercial Applications For Small Business Computers

- General Ledger
- Accounts Receivable
- Inventory
- Job Costing
- Payroll
- Property Management
- Micrograph
- Law Office Acct.

BEACON SOFTWARE INC.

705 Progress Avenue, Unit 17,
Scarborough, Ontario M1H 2X1
(416) 431-3200

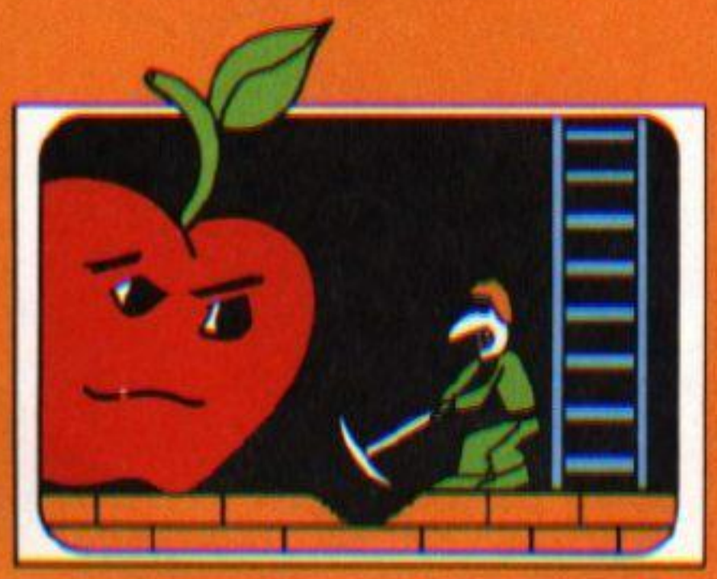


Featuring: MICROGRAPH

The Micrograph systems combines flexibility, accuracy and easy data entry, to provide the most comprehensive business graphic packages available.

VIC-20 ?

SERPENTINE



APPLE PANIC

CHOPLIFTER

VIDEO MANIA

and these are just the games!

TERRAGUARD

ASTROBLITZ

TRASHMAN

BLACK HOLE

CREATIVE SOFTWARE

Write today for our latest catalogue.

ADVANTAGE Computer Accessories

1020 Meyerside Dr., Units 7 & 8, Mississauga, Ontario L5T 1J4 (416) 676-1200

Commodore VIC-20 and Commodore 64 are trademarks of Commodore Business Machines.

NEW!
Commodore 64™
Software & Accessories

We now have one of the largest selections of C64 software & accessories!

BUSINESS

PCS-6480 80-Column Board With a top-rated wordprocessor Includes Data Base and Spreadsheet all linked together – package price	$395.00
WordPac-wordprocessor	$99.95
InquirePac – Data Base	$99.95
CalcPac – Spreadsheet	$99.95
FilePac – Mailing List	$79.95
AccountPac – Home/Bus	$69.95
General Ledger	$95.00

EDUCATIONAL

C64 BASIC Tutorial	$39.95
C64 Tour – Overview	$15.95
Happy Tutor – Typing	$29.95

GAMES

Cyclons – bestseller!	$39.95
Skiman – slalom game	$29.95
Sluggo – boxing game	$29.95
Casino Pac – 3 games	$39.95

UTILITIES

Editor Pac – Prog. Aid	$89.95
Assembler Pac	$89.95
PetSpeed 64 – Compiler	$199
PetSpeed 4000/8000	$249

ACCESSORIES

80-Column Board	$279
RS232 Printer Interface	$129
Parallel Printer Interface	$129
Z-80 Board – runs 40 AND 80 Column CP/M	$399
VIC 20 / C64 Interface IEEE & RS232, totally transparent. Interpod	$245

MODEM For VIC 20 / C64

Accoustic coupled; one cable to computer supplies all signals and power; capable of 110 to 300 baud, full & half duplex - originate. Assembled & tested, with software but without case Only $129

Mail Order Information:

All prices in Canadian dollars
Available from your Commodore dealer, or if not, send check or money order (include 7% sales tax) to

COMPUTER WORKSHOPS
465 King Street East #9
Toronto, Ont. M5A 1L6
Phone (416) 366-6192

Dealer Inquiries Invited

NEW
Basic Utility for the Commodore 64
POWER 64

- **easy to learn**
- **easy to use**
- **program faster and more efficiently with better results**
- **MOREPOWER included FREE**

Powerful Programmer's Utility
by Brad Templeton
Manual by Jim Butterfield

$99.95 from your local Commodore dealer.
For your nearest dealer call:
(416) 273-6350
PRO·LINE SOFTWARE
755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO L4Y 4C5

PRO·LINE SOFTWARE

A CANADIAN COMPANY

designing, developing, manufacturing, publishing and distributing microcomputer software

DEALER ENQUIRIES WELCOME
AUTHOR'S SUBMISSIONS INVITED
CALL OR WRITE

(416) 273-6350
PRO·LINE SOFTWARE
755 THE QUEENSWAY EAST, UNIT 8,
MISSISSAUGA, ONTARIO L4Y 4C5

WICO

COMMAND CONTROL™

Arcade challenge, accuracy, precision, speed, durability, quality, sensitivity and excitement.

Now yours, at home.

Who else but WICO could make that claim? WICO is the world's largest designer and manufacturer of controls for the arcade.

Write today for our latest catalogue.

ADVANTAGE Computer Accessories

1020 Meyerside Dr., Units 7 & 8, Mississauga, Ontario L5T 1J4 (416) 676-1200

Software first, computer second.

Workhorse solutions for tough questions.

OK. You're ready to buy a computer. Here's how to make an intelligent business decision.

Decide on your software first. No computer is better than the software that runs the operation. No software is better than **Southern Solutions'.** We have real business accounting and record keeping software that is right for today's business world. It runs on the best line of computers available today: Commodore including the exciting new Commodore 64™.

Compare our software solutions with others:

FileGuard™ protects your files from loss even by power failure.

SuperMath™ meets your needs as your business expands. Our software with SuperMath will handle numbers up to $1 billion. Most micros stop at far less.

You can design your Balance Sheet, P&L, Budget Analysis, etc.

Complete Systems or Individual Modules handle general ledger, accounts receivable and payable, billing, payroll, mailing lists, oil accounting, pharmacy management, encumbrance accounting, etc.

Our software uses practically any printer and grows as your needs expand.

Real business software for real business computers.
Capability you need at prices you can afford.

Your professional computer dealer can help you make a computer become a productivity tool. For a demonstration, visit

Distributed in Canada by: Canadian Micro Distributors Ltd*
500 Steeles Avenue
Milton, Ontario
L9T 3P7

W E

Southern Solutions

PO. Box 'P'
McKinney, Texas
75069
(214) 542 – 0278

Now For The Commodore 64

* A Division of BMB CompuScience Ltd.

"Commodore 64 is a trademark of Commodore Business Machines, Inc."

CBM INTERFACES

The Connecting Links

Increase Your Computer's Ability

CBM PRINTER ADAPTERS
—addressable-switch selectable upper/lower, lower/upper case
—works with BASIC, WORDPRO, VISICALC and other software
—IEEE card edge connector for connecting disks and other peripherals to the PET®
—power from printer unless otherwise noted

RS-232 SERIAL ADAPTER — baud rates to 9600 — power supply included
MODEL-ADA 1450a $149.00

CENTRONICS/NEC PARALLEL ADAPTER — Centronics 36 pin ribbon connector — handles graphics
MODEL-ADA 1800 ..... $129.00

COMMODORE 64™ to RS-232 CABLE ADAPTER
—plugs into RS-232 port — provides voltage conversions to drive standard RS-232 printers, terminals and mainframes — 6 foot cable included — receives power from computer — uses address #2
—electronics fully enclosed — case 2¾ x 2 inches
MODEL ADA 6410F (Female Connector) $79.00
MODEL ADA 6410M (Male Connector) $79.00
MODEL ADA 64115 Modem Cable $79.00

COMMUNICATIONS ADAPTER — serial & parallel ports — true ASCII conversion — baud rates to 9600 — half or full duplex — X-ON, X-OFF — selectable carriage return delay — 32 character buffer — centronics compatible — power supply included
MODEL SADI $295.00

ANALOG TO DIGITAL CONVERTER — 16 channels — 0 to 5.12 volt input voltage range — resolution 20 millivolts per count — conversion time less than 100 microseconds per channel
MODEL-PETSET1 ..... $295.00

US Dollars Quoted
$5.00 Shipping & Handling
MASTERCHARGE / VISA

IN THE USA order from:
Connecticut microComputer, Inc.
36 Del Mar Drive
Brookfield, CT 06804
(203) 775-4595 TWX: 710 456-0052

IN CANADA order from:
Batteries Included, Ltd.
186 Queen Street West
F6 Toronto, Canada M5V 1S1
(416) 596-1405

Dealer Inquiries Invited

NEW

Assembler for the Commodore 64

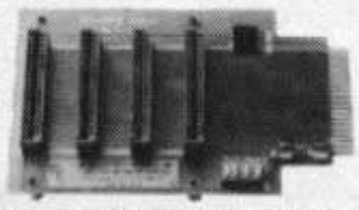
PAL64

- **easy to learn**
- **easy to use**
- **fast**
- **comprehensive manual**

Personal assembly language
by Brad Templeton
also available for the Commodore
4,000 - 8,000 - 9,000 series

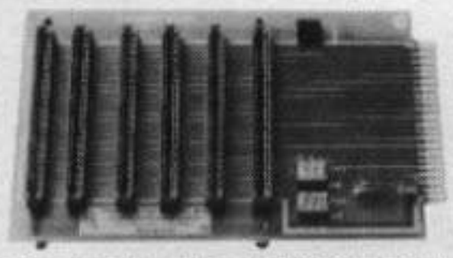
$99.95 from your local Commodore dealer.
For your nearest dealer call:
(416) 273-6350
PRO-LINE SOFTWARE
755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO L4Y 4C5

VIC-20 and CBM 64 EXPANDER BOARDS

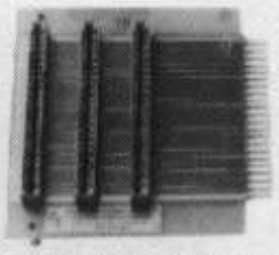


4 Slot for the 64. Toggle switches and reset switch.
P/N C64 **$69.95**

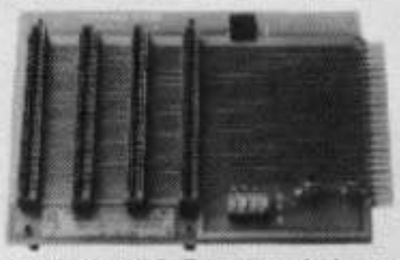
PTI offers the finest selection of expander boards available for the VIC-20 and CBM 64. The design features, quality construction, and competitive prices make any of them an exceptional value. New products are being added monthly, so write for complete catalog.



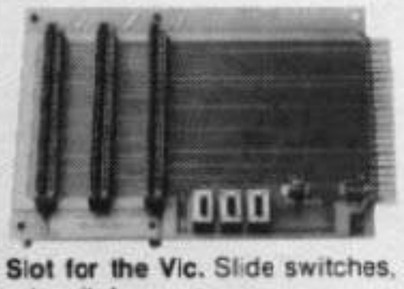
6 Slot for the VIC. Toggle switches and reset switch.
P/N V36 **$79.95**



Slot for the VIC. No switches, reset, or fuse.
P/N V13 **$49.95**



4 Slot for the VIC. Toggle switches and reset switch.
P/N V24 **$69.95**



3 Slot for the Vic. Slide switches, no reset switch.
P/N V23 **$59.95**

PRECISION TECHNOLOGY, INC.
COMPUTER PRODUCTS DIVISION
P.O. BOX 15454
SALT LAKE CITY, UTAH 84115
(801) 487-6266

See your dealer, or place your order direct

VISA-M/C-CHECK-COD

Advertising Index

Software

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Advantage Computer Acc.	VIC 20 games				**75**			
Beacon Software	Business Packages				**74**			
Computer Marketing \Canadian Micro	Calc Result (spreadsheet prog.)	Handic Software ab		IBC	**IBC**			
Micro Applications	Master (programming aid)		IBC					
Pacific Coast Software	C64: Utility, Business, Games			62				
Precision Software	Superscript (wordprocessor)		61					
Pro-Line Software	PAL 64 (assembler)				**79**			
	POWER 64 (programming aid)				**76**			
Southern Solutions	Business packages				**78**			
Wycor Business Systems	Provincial Payroll			64				

Hardware

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Computer Workshops \Computer Marketing	Z-RAM (CP/M board)	Madison		63				
Connecticut microComputer	PET/CBM Interface adapters			62	**79**			
Precision Technology	VIC20/C64 Expander Boards			61	**79**			
Richvale Telecommunications	C64 Link (IEEE adapter)			IFC	**IFC**			

Accessories

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Advantage Computer Acc.	Joysticks	Wico			**77**			
Computer Workshops	Apr83 products list				**76**			
Consultors Int'l	Stock Market With Your PC (book)		64					
"	A To Z Book Of Games		64					
"	Dynamics Of Money Mgmt (book)		64					
"	Invment. Analysis w/Your Micro (book)		64					
Leading Edge Inc.	Elephant Diskettes		BC	BC	**BC**			
Toronto PET Users Group	Membership info			61				

A More Powerful Planning
And Forecasting Tool Than Any Other On The Market . . .

Calc Result

A three-dimensional spread sheet
with multiple pages of 63 x 254 cells
which utilizes only the memory
in cells that are active

Produces Graphics (Histograms) on screen
and printer

Gives unlimited possibilities in each cell with
IF-THEN-ELSE with AND, OR and NOT-ELSE

Available now for 8032 and 8096 - *coming soon* for Commodore 64.

FOR FURTHER INFORMATION CONTACT YOUR NEAREST COMMODORE DEALER

OR

IN CANADA

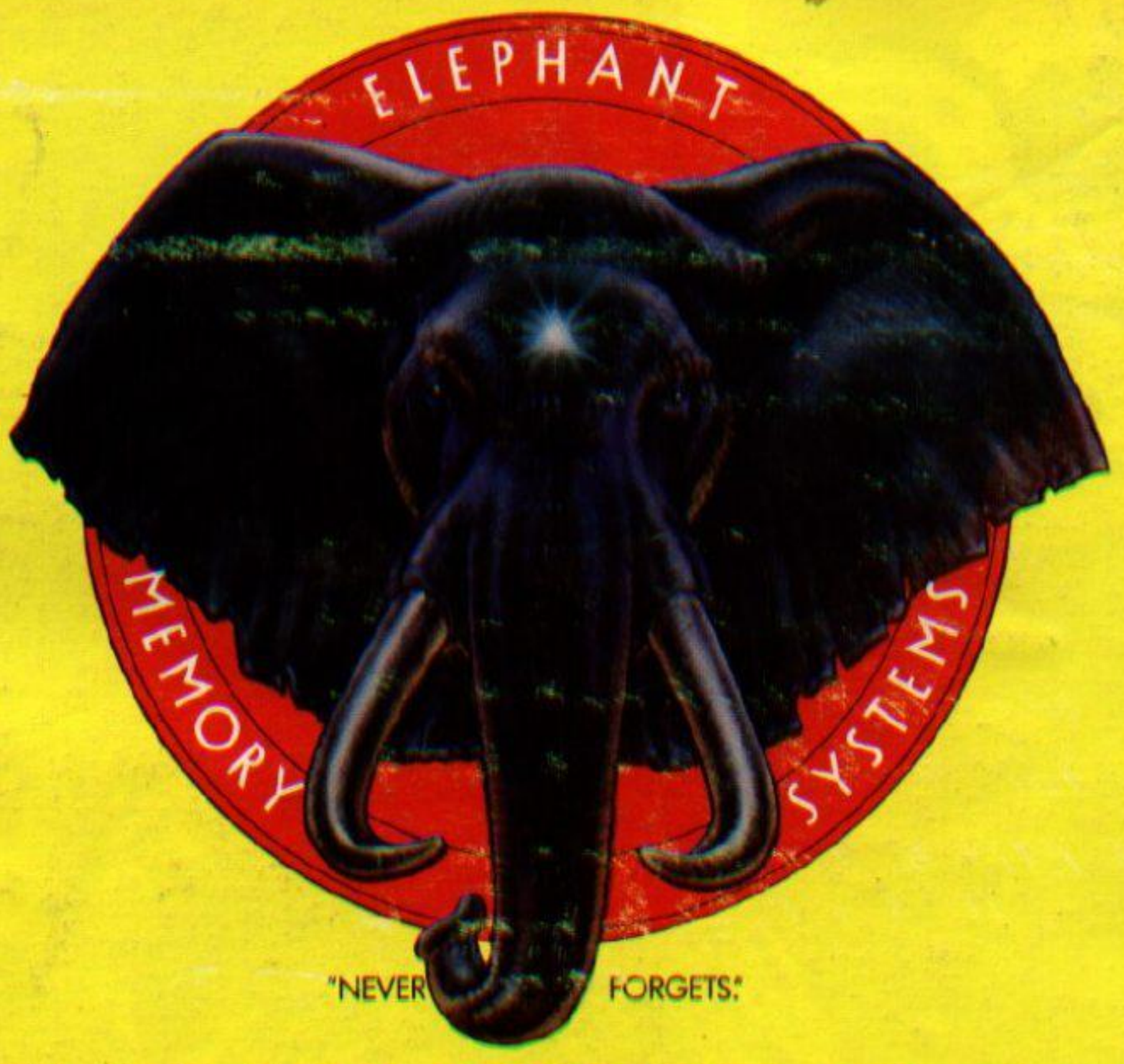
CANADIAN MICRO DISTRIBUTORS
500 Steeles Avenue
Milton, Ontario, Canada L9T 3P7
(416) 878-7277

IN THE U.S.A.

COMPUTER MARKETING SERVICES INC.
300 West Marlton Pike, Suite 26
Cherry Hill, New Jersey, U.S.A. 08002
(609) 795-9480

CALC RESULT *is a trademark of Handic Software ab*

REMEMBER:



"NEVER FORGETS."

MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021

Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.